COLLABORATIVE GAMES

REFERENCE MANUAL v1.4

31 October 2019

Index

HOW IT WORKS **5**

GAME ACTIVATION **6**

GAME DESCRIPTION **7**

VOICE Parameter 10

Language Manager Parameter **11**

Package Manager **12**

Background Scripts (advanced mode W.I.P.) **13**

you can develop scripts that work the background over other scripts, even if they have a higher priority. 13

How can they be used? 13

Game Failed **15**

Game Map Mode **16**

TRIAL DESCRIPTION **17**

TRIAL TYPE **19**

type: “photo” 19

type: “photoselfie” 20

type: “video” 21

type: “link” 22

type: “info” 22

type: “qrcode” 23

type: “qrcodepair” 24

type: “qrcodecollect” 25

type: “unused” 28

type: “gpsposition” 29

type: “quiz” 30

type: “quizopen” 31

type: “quizphoto” 32

type: “puzzle” 33

type: “puzzleslider” 34

type: “photo360” 35

type: “object3d” 36

type: “bluetooth” 37

Turn Bluetooth on or off 37

Step 1: Pair 37

Step 2: Connect 37

type: “arduino” 39

Trials Type Advanced (only usable with code) **40**
- Near-field communication (NFC) 40
- type: “nfcread” 40
- iBeacon 40
- type: “ibeacon” 41

Proximity Sensors 42

NFC ADVANCED DEVELOPMENT 43
- Near-field communication (NFC) 43
- Near-field communication (NFC) Command 43
  - Execute a trial 43
  - Activate a trial 43
  - Failed a trial 43
  - Hidden a trial 43
  - Load a scripts 44
  - Load a code and execute 44

CONNECT TO ARDUINO 45

COMMON OPTION 46
- Waiting 46

TRIAL ACTIVATED STRUCTURE 47

TRIAL HIDDEN STRUCTURE 49

SPECIAL COMMANDS 50
- TRIAL QRCODE AUTO-EXECUTE 50

TRIAL AFTER COMPLETE 51

TRIAL AFTER FAILED 53

TRIAL SCORE OPTIONS 54

EDITOR DEBUG MODE 55

ON APP DEBUG 57

EDITOR 58
- BLOCK 58
- BRICK 58

BLOCK & BRICK PHILOSOPHY 60

USE THE WEB EDITOR 62
- Select a script 62
- Select a template 63
- Script Setting 64

Create the trials 65
Define a trial 66
Setting a Brick 67

ADDED A TEMPLATE 68
Added asset 68

ELEMENT DEFINITION 70

CREATE A NEW BLOCK 71

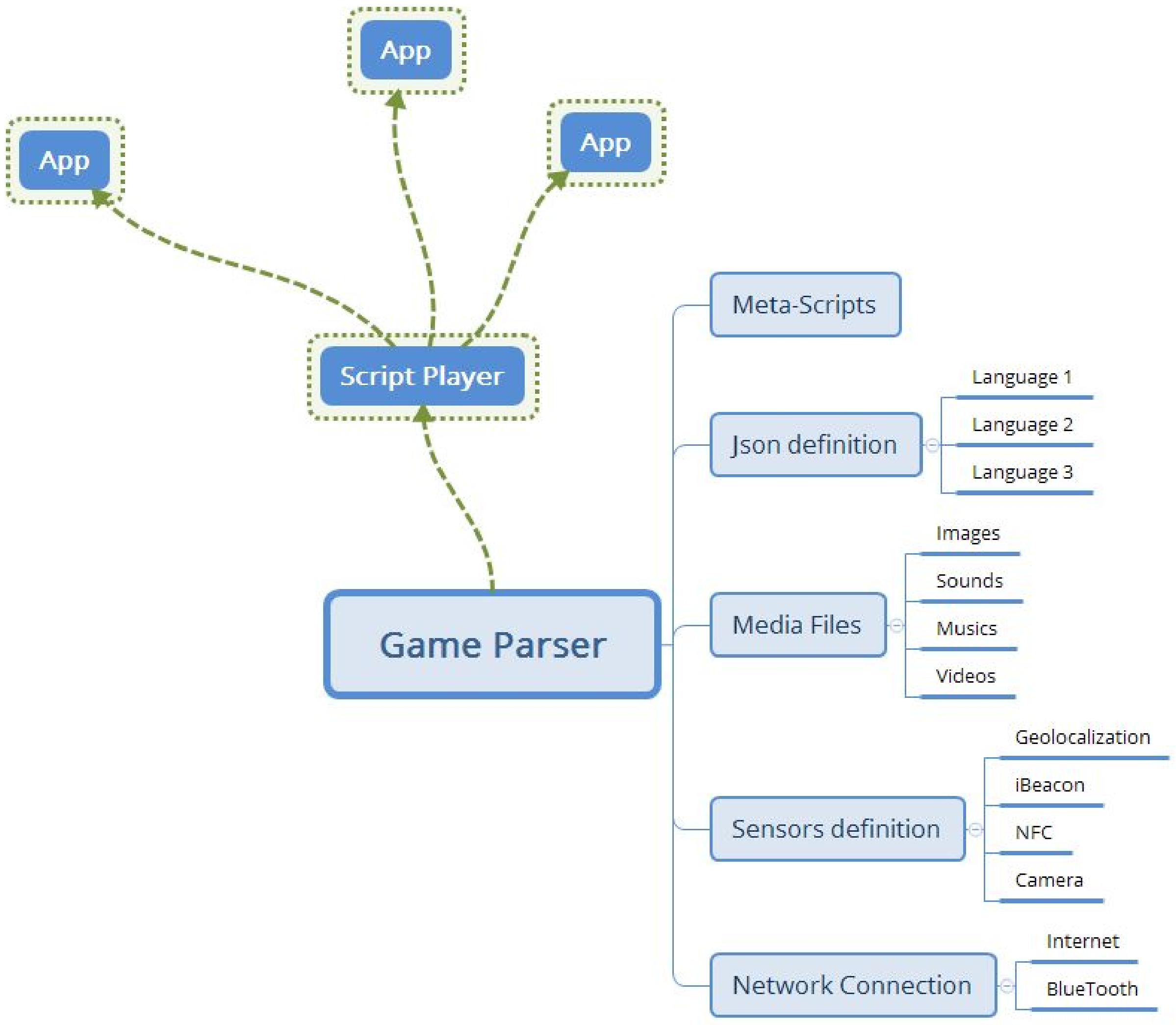
CREATE A NEW BRICK 72
BRICK DEFINITION EXAMPLE 73
Block Definition Example 74
Abstract data management and Polymorphic code 76

EDITOR PRO 83
ToolBar Menu 83
Main Menu 85
GAME SETTINGS 86
TRIAL OPTIONS 88
Internal JSON Script Editor 90

USE THE EDITOR PRO 91
New Script 91
Use Templates 93
Project Parameters 94
Login to PLUGGY 94
Languages (default and included) 95
Create a package 95
Upload a package to PLUGGY 96
External Tool Setting 96

HOW IT WORKS

Games Hunter is a multi-layer system that allows you to run scripts within different applications that are used as "players".
This allows you to develop different scripts that will work in different ways within the same application.



GAME ACTIVATION

There are three possible ways to access the games:

- The list is available and can be downloaded on the Pluggy platform. In this case, it's possible to filter through various parameter (e.g. date, location, name, caption)
- Through geolocation (the ten closest events to the player are shown in the screen)
- Through the QR code, which activates the link and the user can directly download the game. In this case, the QR code can be set either at the museum entry or inside a book, near an artefact where it can be easily found by the user. Once it is activated, the game is downloaded and executed.

The format is @PLUGGYSCRIPT=url

The parameter url represent the link to download the script

GAME DESCRIPTION

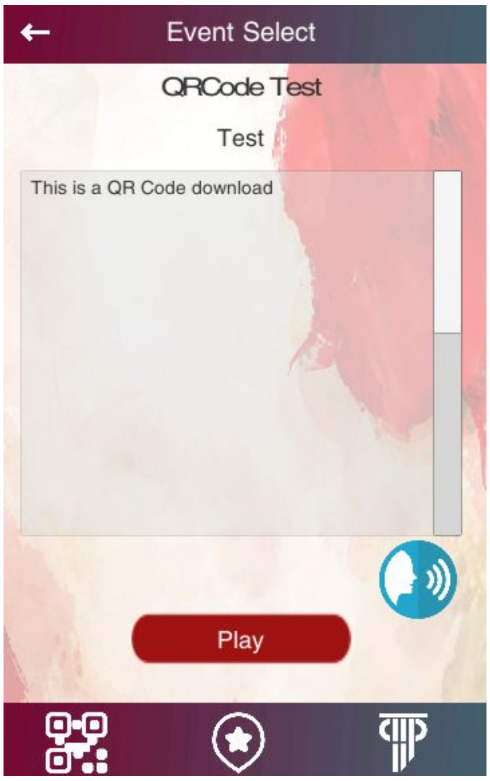
Function	Description	Format	Note
ID	game id	integer	
UUID	game uuid	string	
Enabled	If this game is enabled	bool	
Icon	Icon path	string / url	
Name	name of the game	string	
SubTitle	short subtitle	string (128 char)	
Description	description of the game	string (255 char)	
Duration	duration of the game (minutes)	integer	If “0”: no time limit
Data start	date of enabling	data/string	
Data end	date of disabling	data/string	
Place	place name	string	e.g. Rovigo
GeoCoord	Location Coordinate	lat:float long:float range:float (all this are converted from strings)	Useful to create the list of games near the user Range : distance to activate in auto-mode if is different from 0
Visible	if the game is visible	bool	
Private	if this game is private	bool	Only in the museum (for examples)
Trials	list of trials	json array	
Complete	the result if a game is completed	json object	
Failed	the result if a game is failed	json object	Only with duration != of 0 or if the player leaves the game

Voice	This activates the vocal synthesis	bool	WIP activates a voice with the current culture if is possible and present on the device. To be used with attention.
StartScore	start score	int	start score
Debug	if start for debug mode	bool	if debug
Alone	se viene giocato da solo	bool	if play alone, not a collaborative game
Package	if is a zip internal package for offline mode	bool	
Package_ref_url	reference url for the package	string	
Package_ref_internal	reference internal position for the package	string	
Kiosk	if start like a kiosk / totem	bool	if activate work only in totem mode.
MapMode	if use a map mode	json object	if != null activate the map mode
Created	creation date	string	
Modified	modified date	string	
RunInBackground	if the script run in background	bool	

```
{
  "data": {
    "id": 0,
    "uuid": "",
    "enabled": false,
    "icon": "",
    "name": "",
    "subtitle": "",
    "description": "",
    "duration": 0,
    "datestart": "",
    "dateend": "",
    "place": "Rovigo",
    "geocoord": {
      "latitude": "45.50",
      "longitude": "11.15"
    },
    "visible": false,
    "private": false,
    "trials": [
        ],
“debug”:false,
    "complete": null,
    "failed": null
“startscore”:100,
  }
}
```

VOICE Parameter

The VOICE parameter activates a button which allows the conversion of texts into synthesized audio files (it's the device itself which does the conversion without the aid of any pre-registered audio files e.g. mp3).
The conversion system appears in two sections: when the user chooses the event and when he chooses the trial. Both title and description are converted.
The system adopts the voices embedded on the device and it is able to change between different languages and pronunciations available: English, Italian, Greek and Slovak.



Language Manager Parameter

The language manager allows you to incorporate several languages into a package and thus to change several languages within the project.

The script is loaded with a main language, there is an array with all possible languages (identified by an id) and related scripts.

Conventionally the name of the script identifies the language:

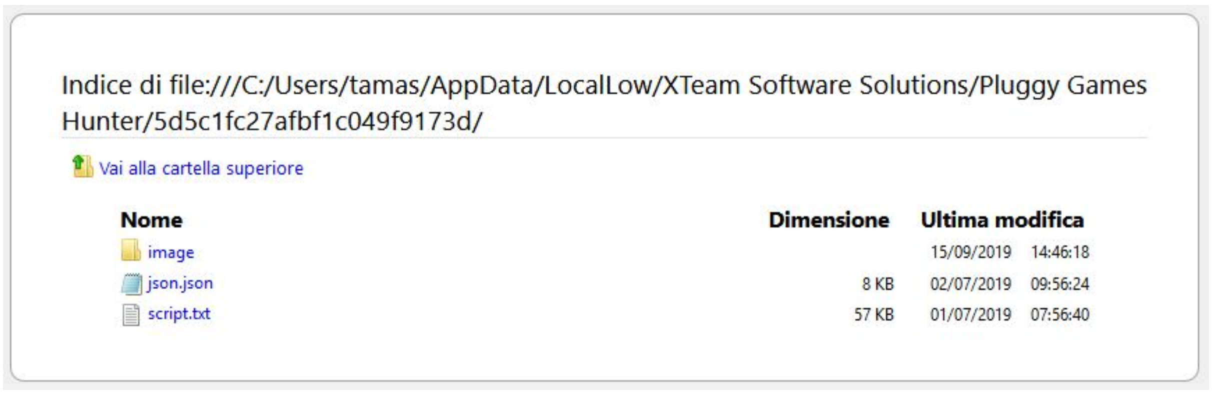
Script name	Language ID	Language
json.json	=	Main language
jsonen.json	“eng”	English language
jsongr.json	“gre”	Greek language
jsonit.json	“ita”	Italian language
jsones.json	“esp”	Spanish language
jsonsk.json	“slo”	Slovak language

NOTE: it would be possible to change parts of the game in relation to the type of language, for example changing the answers of a quiz. This is not recommended, however, it is only possible to do it manually.

OPTIONS

Function	Description	Format	Note
langid	The languageID	string	
script	a link to the script with the language	string	

Package Manager



Indice di file:///C:/Users/tamas/AppData/LocalLow/XTeam Software Solutions/Pluggy Games Hunter/5d5c1fc27afbf1c049f9173d/

Vai alla cartella superiore

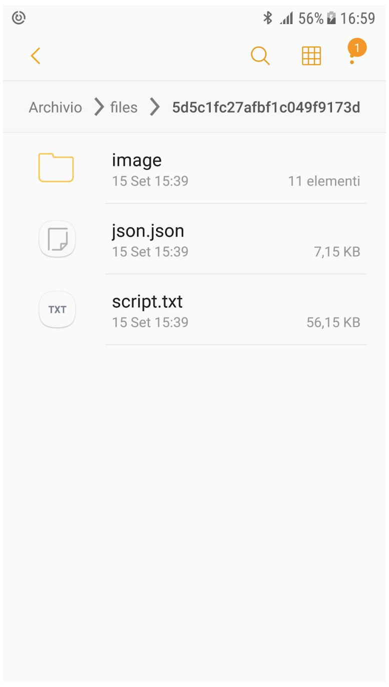
Nome	Dimensione	Ultima modifica
image		15/09/2019 14:46:18
json.json	8 KB	02/07/2019 09:56:24
script.txt	57 KB	01/07/2019 07:56:40

is a system that allows you to create packages containing the data of a game / event that will be used in Games Hunter.

The package (zip) contains all the data necessary to execute the script:

- meta-script
- json
- languages
- images
- videos
- sounds/music

The package can be downloaded from different web locations (Pluggy web site) or installed manually.



56% 16:59

Archivio > files > 5d5c1fc27afbf1c049f9173d

image	15 Set 15:39	11 elementi
json.json	15 Set 15:39	7,15 KB
script.txt	15 Set 15:39	56,15 KB

Background Scripts (advanced mode W.I.P.)

you can develop scripts that work the background over other scripts, even if they have a higher priority.

You need to enable the json **RunInBackground** to access this feature.

These scripts (you can run multiple background scripts at the same time even if it is not a recommended operation) remain running regardless of the operations performed by the application.

How can they be used?

The fundamental purpose is to have an interaction of the application in which they are running independent of the main script (if running at that time).
So it is possible for the application to run a game and, for example, if the user reaches a particular area, the script in the background will run a trial linked to that geolocalized location (for example listening to an audio explanation of a place).

**WARNING. Pay a lot of attention in developing scripts that work in the background, scripts don't have to intertwine commands and must allow foreground scripts to be executed without any kind of constraint or issues.
(for example avoid to create 2 geolocalized trials in the same place in 2 different scripts)**

Game Complete

This structure is used when the game is complete. The info should be captivating and engaging. Is possible to show a prize or a reward to the user.

OPTIONS

Function	Description	Format	Note
info	a description for the game complete, show on feedback screen	string	
link	a link to redirect the user (for examples for a prize)	string	
photo	a link to a photo	string	
sound	a link to a sound	string	mp3
score	score to added	integer	

Game Failed

This structure is used when the game is failed.

OPTIONS

Function	Description	Format	Note
info	a description for the game complete, show on feedback screen	string	
link	a link to redirect the user	string	
photo	a link to a photo	string	
sound	a link to a sound	string	mp3
score	score to added	integer	

Game Map Mode

This structure is used when the game use a maps (an image) and POIs (point of interest) to execute the trials.

OPTIONS

Function	**Description**	**Format**	**Note**
photo	a link to a photo	string	
sound	a link to a sound	string	mp3
pois	list of json with the pois	json object array	int x int y int trialid

TRIAL DESCRIPTION

Each game consists of a list of trials, every trial has a name and a description. Each type of trials has specific purposes which can be interchanged through different ways to create stories.

Function	Description	Format	Note
ID	trial id	integer	
Type	type of trial	string	see below
Name	trial’s name	string	
Description	trial’s description	string	
Score	a score of the trials if completed	integer	
activated	activated structure	json object	see below
hidden	hidden structure	json object	see below
options	option structure	json object	see below
aftercomplete	aftercomplete structure	json object	see below
afterfailed	afterfailed structure	json object	see below (not used with editor)
scoreoptions	scoreoptions structure	json object	see below (not used with editor)

```
{
  “id”: 0,
  "type": “”,
  "name": "",
  "description": "",
  "score": 10,
  "activated":{
      },
 "hided":{
      },
  "options": {
      },
  "aftercomplete": {
  },
 "scoreoptions": {
  }

}
```

TRIAL TYPE

- type: “photo”

This trial let the user take pictures. More specifically, the user is asked to take a photo of something described in the trial task. The description should be captivating and engaging.

This trial activates the camera directly, to provide information on the task enter a trial info before this trial

OPTIONS

Function	Description	Format	Note
share_facebook	The trial is complete only after sharing the photo to facebook	boolean	
share_twitter	The trial is complete only after sharing the photo to Twitter	boolean	
share_instagram	The trial is complete only after sharing the photo to Instagram	boolean	
share	The trial is complete only after sharing the photo (facebook or twitter or)	boolean	
share_optional	if the share is facultative and not mandatory	boolean	
share_score	Point added if shared	int	

```
{
    "share_facebook": false,
    "share_twitter": false,
        "share_instagram": false,
    "share": "false"
  }
```

- type: “photoselfie”

This trial let the user take pictures. More specifically, the user is asked to take a selfie of something described in the trial task. The description should be captivating and engaging.

This trial activates the camera directly, to provide information on the task enter a trial info before this trial

OPTIONS

Function	Description	Format	Note
share_facebook	The trial is complete only after sharing the photo to facebook	boolean	
share_twitter	The trial is complete only after sharing the photo to Twitter	boolean	
share_instagram	The trial is complete only after sharing the photo to Instagram	boolean	
share	The trial is complete only after sharing the photo (facebook or twitter or)	boolean	
share_optional	if the share is facultative and not mandatory	boolean	
share_score	Point added if shared	int	

```
{
    "share_facebook": false,
    "share_twitter": false,
        "share_instagram": false,
    "share": "false"
  }
```

- type: “video”

In this case, the trial makes a video of something like suggesting the description of the trial. It is better if is it’s something's pleasant.

This trial activates the camera directly, to provide information on the task enter a trial info before this trial

OPTIONS

Function	Description	Format	Note
share_facebook	the trial is complete only after sharing the video to facebook	boolean	
share_twitter	the trial is complete only after sharing the video to twitter	boolean	
share_instagram	the trial is complete only after sharing the photo to Instagram	boolean	
share	the trial is complete only after sharing the video	boolean	
share_optional	if the share is facultative and not mandatory	boolean	
share_score	Point added if shared	int	

```
{
    "share_facebook": false,
    "share_twitter": false,
        "share_instagram": false,
    "share": "false"
  }
```

- type: “link”

In this case, the trial opens a link with a browser (it is possible to attach sites link, info.,.)

OPTIONS

Function	Description	Format	Note
url	link url	string	
url_photo	link to a photo	string	
url_sound	a link to a sound	string	mp3

- type: “info”

In this case, the trial consists of reading the text. It’s possible to use this type of trial to give some additional information or to create storytelling between 2 trials.

OPTIONS

Function	Description	Format	Note
url_photo	link to a photo	string	
url_sound	a link to a sound	string	mp3

- type: “qrcode”

In this case, the scope of the trial is to find a QR code.
The QR code needs to start with @PLUGGYQRCODE=word

@PLUGGYQRCODE=gioconda

If “complete” is set, this means that all trials with the word “gioconda” are completed.

This trial activates the camera directly, to provide information on the task enter a trial info before this trial

OPTIONS

Function	Description	Format	Note
word	word to find	string	

@PLUGGYQRCODE=gioconda

- type: “qrcodepair”

In this case, the scope of the trial is to find 2 QR Codes (like a memory game).
The QR code needs to start with
@PLUGGYQRCODEPAIR1=word / @PLUGGYQRCODEPAIR2=word

@PLUGGYQRCODEPAIR1=gioconda
@PLUGGYQRCODEPAIR2=gioconda

At the first QR Code Enabled, the screen displays the suggestion to find another QR Code with Gioconda.
At the second QR Code completed, the trial is completed.

This trial activates the camera directly, to provide information on the task enter a trial info before this trial

OPTIONS

Function	Description	Format	Note
word	word to find	string	

@PLUGGYQRCODEPAIR1=gioconda
@PLUGGYQRCODEPAIR2=gioconda

- type: “qrcodecollect”

In this case, the scope of the trial is to collect words with the use of QR codes.
The QR Code needs to start with @PLUGGYCOLLECTED=id, word, word, word

@PLUGGYCOLLECTED=3, picture, oil, da vinci

The id (e.g:3) identifies the unique trial’s id in order to avoid duplicate QR Code Collected

OPTIONS

Function	**Description**	**Format**	**Note**
word	words to collect	string	NOTE: In order to collect the same word, it is necessary only to repeat the same word. example: oil, oil, oil. This means that one needs to collect 3 “oil” to complete the trail
time	Committed time to complete the trial	integer	This number represents the minute from the start of the game. If "0" appears it means that the time to complete the trial is unlimited. If time is up but the trial is not completed yet, the trial changes status to "invisible"

@PLUGGYCOLLECTED=1, hello, cat,dog

@PLUGGYCOLLECTED=2, hello, ciao

@PLUGGYCOLLECTED=3, cat

- type: “unused”

This trial’s type is made to remove a trial from the game, is set from the application, to remove a trial from the game.
If a trial is not completed, it changes the type to “unused”, so it not considered anymore. This is especially true for those trials with the timer: indeed, when the time stops and the trial is not completed yet, is become unused.

OPTIONS

Function	Description	Format	Note

- type: “gpsposition”

In this case, the trial will be completed if the user approaches to the GPS position. Trough this trial, it is possible to set games around the city or in points of interest.

OPTIONS

Function	Description	Format	Note
gps	geolocalization position and range	object json	range in meters, we suggest 10mt-25mt

```
“gps”:{
“latitude”:”45”,
“longitude:”11”,
“range”:”3”
}
```

- type: “quiz”

In this case, the trial activates a quiz with some closed answers.

OPTIONS (QUIZ JSON OBJECT)

Function	Description	Format	Note
photo	photo to use/download	string	
sound	a link to a sound	string	
question	question	string	
answer	list of answer	strings array max 8 answers, max 4 answer showed, the correct is the first on the list.	

```
“quiz”:{
“photo”:”url”,
“question:””,
“answer”:{””,””,””}
}
```

- type: “quizopen”

In this case, the trial activates a quiz with a open answers.

OPTIONS (QUIZ JSON OBJECT)

Function	Description	Format	Note
photo	photo to use/download	string	
sound	a link to a sound	string	
question	question	string	
answer	list of answer	strings array use multi “right” answer	
casesensitive	if is case sensitive	boolean	

```
“quiz”:{
“photo”:”url”,
“question:””,
“answer”:{“”,””,””},
“casesensitive”:false
}
```

- type: “quizphoto”

In this case, the trial activates a photo quiz with some closed answers (photos).

OPTIONS (QUIZ JSON OBJECT)

Function	Description	Format	Note
photo	photo to use/download	string	
sound	a link to a sound	string	
question	question	string	
answer	list of url	strings array with url, is possible insert 8 answer but only 4 are showed	

```
“quiz”:{
“photo”:”url”,
“question:””,
“answer”:{”url”,””,””}
}
```

- type: “puzzle”

The trial activates a puzzle of images.

OPTIONS (PUZZLE JSON OBJECT)

Function	Description	Format	Note
photo	photo to use/download	string	
pieces	number of pieces	string	9/16 pieces, number

```
“puzzle”:{
“photo”:”url”,
“pieces:””
}
```

- type: “puzzleslider”

The trial activates a puzzle slider of images.

OPTIONS (PUZZLE JSON OBJECT)

Function	Description	Format	Note
photo	photo to use/download	string	
pieces	number of pieces	string	9/16 pieces, number

“puzzle”:{
“photo”:”url”,
“pieces:””
}

- type: “photo360”

The trial activates a 360 photo, only for view

OPTIONS (JSON OBJECT)

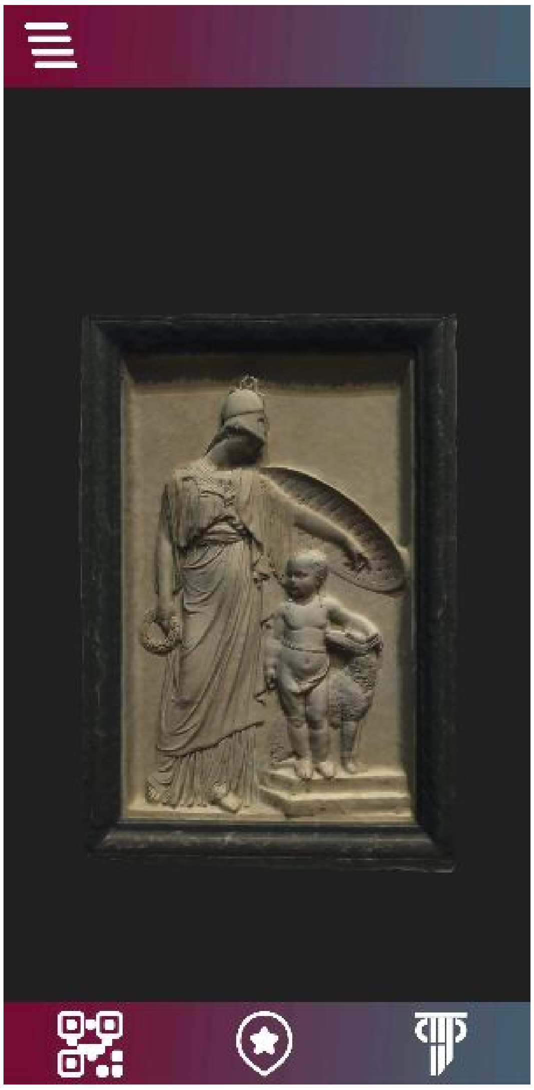
Function	Description	Format	Note
url_photo	url photo to use/download (360 photo)	string	

- type: “object3d”

The trial activates a 3d object, only for view.
The user can manipulate the rotation and the size with the finger gesture.

OPTIONS (JSON OBJECT)

Function	Description	Format	Note
url	url to use/download the 3d object (fbx)	string	



- type: “bluetooth”

The trial connect to a bluetooth to receive a message. The message can activate / execute a different trial

OPTIONS (JSON OBJECT)

Function	Description	Format	Note
command	command to interact with the trial	string	

Before communicating with the Bluetooth device run the pair.

Turn Bluetooth on or off

- Open your device's Settings app.
- Tap **Connected devices** › **Connection preferences** › **Bluetooth**.
- Turn **Bluetooth** on or off.

Pair & connect a Bluetooth accessory

Before you can connect to a Bluetooth accessory, you must pair it with your device. After pairing, your devices stay paired until you unpair them.

Step 1: Pair

1. Open your device's Settings app.
2. Tap **Connected devices** › **Connection preferences** › **Bluetooth**. Make sure Bluetooth is turned on. If you don't see "Connection preferences," go to Step 3.
3. Tap **Pair new device**. If you don't see "Pair new device," you're using a different Android version. Look under "Available devices." If needed, tap More ⋮ › **Refresh**.
4. Tap the name of the Bluetooth device you want to pair with your phone or tablet.
5. Follow any on-screen steps.

Tip: If you need a passcode and don't have it, try 0000 or 1234 (the most common passcodes).

Step 2: Connect

1. Open your phone's Settings app.
2. Tap **Connected devices** › **Connection preferences** › **Bluetooth**. If you don't see "Connection preferences," go to Step 3.
3. Make sure **Bluetooth** is turned on.
4. In the list of paired devices, tap a paired but unconnected device.
5. When your phone and the Bluetooth device are connected, the device shows as "Connected."

- type: “arduino”

The trial connect to an Arduino with bluetooth to receive a message (only support HC-05 / HC-06 / HC-08). The message can activate / execute a different trial

OPTIONS (JSON OBJECT)

Function	Description	Format	Note
command	url to use/download the 3d object (fbx)	string	

Trials Type Advanced (only usable with code)

Near-field communication (NFC)

Near-field communication (NFC) is a set of communication protocols that enable two electronic devices, one of which is usually a portable device such as a smartphone, to establish communication by bringing them within 4 cm of each other.

NFC tags contain data and are typically read-only. They can be custom-encoded by their manufacturers or use NFC Forum specifications. The tags can securely store personal data such as debit and credit card information, loyalty program data, PINs and networking contacts, among other information.

- ### type: “nfcread”

The aim of the trial is to find a NFC.
The NFC message needs to start with @PLUGGYNFCREAD=word

@PLUGGYNFCREAD=gioconda

set “complete” all the nfc trials with word equal to “gioconda”

OPTIONS

Function	Description	Format	Note
word	word to find	string	

iBeacon

iBeacon is based on Bluetooth low energy proximity sensing by transmitting a universally unique identifier picked up by a compatible app or operating system. The identifier and several bytes sent with it can be used to determine the device's physical location, track customers, or trigger a location-based action on the device such as a check-in on social media or a push notification.

iBeacon can also be used with an application as an indoor positioning system, which helps smartphones determine their approximate location or context. With the help of an iBeacon, a smartphone's software can approximately find its relative location to an iBeacon in a store. Brick and mortar retail stores use the beacons for mobile commerce, offering customers special deals through mobile marketing, and can enable mobile payments through point of sale systems.

Another application is distributing messages at a specific Point of Interest, for example a store, a bus stop, a room or a more specific location like a piece of furniture or a vending machine. This is similar to previously used geopush technology based on GPS, but with a much reduced impact on battery life and better precision.

- ## type: “ibeacon”

The aim of the trial is to find an iBeacon (or compatible system).
It is possible to identify any ibeacon with the UUID (Universally Unique Identifier).
If you set “complete” all the trials with the required UUID become completed.

OPTIONS

Function	Description	Format	Note
UUID	iBeacon UUID	string	
range	range to activate	integer	value in cm

Proximity Sensors

W.I.P.

NFC ADVANCED DEVELOPMENT

Near-field communication (NFC)

As previously written Near-field communication (NFC) is a set of communication protocols that enable two electronic devices, one of which is usually a portable device such as a smartphone, to establish communication by bringing them within 4 cm of each other.

NFC tags contain data and are typically read-only. They can be custom-encoded by their manufacturers or use NFC Forum specifications. The tags can securely store personal data such as debit and credit card information, loyalty program data, PINs and networking contacts, among other information.

Near-field communication (NFC) Command

Execute a trial

```
@PLUGGYNFCEXECUTE=id:trial_id,uuid:event_uuid

@PLUGGYNFCEXECUTE=id:1

@PLUGGYNFCEXECUTE=id:1,uuid:1000
```

Activate a trial

```
@PLUGGYNFCACTIVATE
```

Failed a trial

```
@PLUGGYNFCFAILED
```

Hidden a trial

```
@PLUGGYNFCHIDDEN
```

Load a scripts

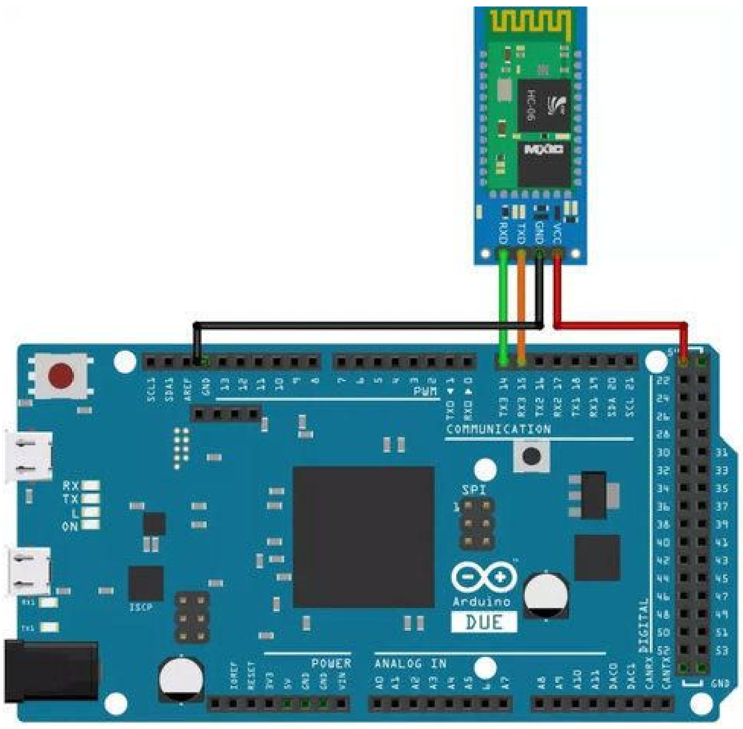
@PLUGGYNFCSCRIPT= Url

Load a code and execute

@PLUGGYNFCCODE=Json code

CONNECT TO ARDUINO

Through the bluetooth connection you can communicate with an Arduino (developing the appropriate script in Arduino IDE)



It will thus be possible to send commands in relation to the different types of sensors that can be connected to arduino.

This allows to create interactive artistic installations.

COMMON OPTION

- Waiting

OPTIONS

Function	Description	Format	Note
wait_timer	timer in seconds	integer	Then, activate the execute on the aftercomplete section (if it is not present, one should return to the trial’s list)
wait_click	wait for a click	boolean	Then, activate the execute on aftercomplete section (if it is not present, one should return to the trial’s list)

TRIAL ACTIVATED STRUCTURE

This structure describes which trials are activated.
If the structure is null, one should not start, instead, he should wait to activate from another trial or other events.

Function	Description	Format	Note
invisible	If it is true, this trial can't be visible to the user	boolean	
onstart	It is activated at the start of the game	boolean	
ontimer	It is activated after a few minutes from the start of the game	integer	
onscore	It is activated after the user collected points => of this value	integer	
onqrcode	if the user collects a special QR code, this will start with @PLUGGYACTIVATE= word	string	This is the word that activates the trial: example QRCode @PLUGGYACTIVATE= example
ongps	activate if the user is near to a gps location	json object	latitude longitude range
onibeacon	activate is the user is near to an ibeacon	json object	

```
{
    "onstart": true,
    "ontimer": 0,
    "onscore": 0,
      "onqrcode": "",
“ongps”:{
“latitude”:”45”,
“longitude:”11”,
“range”:”3”
}
    }
```

TRIAL HIDDEN STRUCTURE

This structure describes in which ways the trials are hidden (deactivated) if not completed.
If the structure is null, do not deactivate the trial.

Function	Description	Format	Note
oncomplete	It is deactivated if another trial has been completed before this one.	string	WIP The next functionality will allow executing some boolean operations. E.g. If trial #1 and #2 are completed (trial 1 && trial 22), turn off trial #5 (|| trial 5)
ontimer	It is deactivated after a few minutes from the start of the game	integer	After a certain amount of time, the trial is hidden
ontimerfromactivate	It is deactivated after a few minutes from the activate of the trial.	integer	
onscore	It is deactivated after the user collected points => of this value	integer	After reaching a precise score, the trial is deactivated

ongps	It is deactivated if the user is near a gps location	json object	“ongps”:{
			“latitude”:”45”,
			“longitude:”11”,
			“range”:”3”
			}

```
 {
     "oncomplete": “”,
     "ontimer": 0,
     "onscore": 0,
“ongps”:{
“latitude”:”45”,
“longitude:”11”,
“range”:”3”
}

   }
```

SPECIAL COMMANDS

TRIAL QRCODE AUTO-EXECUTE

It is possible to execute a trial through a QRcode which needs to contain the command @PLUGGYEXECUTE=name

Where the parameter “name” is simply the trial name.
The requirement is that the trial name must be unique. Otherwise, the first trial will be activated with that name, even if it is not yet complete.

This previous feature could be also adopted to create different trials with the same name, if, for instance, the goal is to execute them randomly. This feature has not been tested yet.

TRIAL AFTER COMPLETE

This structure describes some action after a trial is complete.

Function	Description	Format	Note
more info	This permit to add a string with suggestions, feedbacks, engage messages	string	
eventcompleted	This complete the event, used to complete an event	boolean	if true the event is completed
activate	This activates the following next trial ID to activate in the list of trials	string	NOTE: in next version it will be possible to insert more ID separated with a comma id1,id2,id3
hidden	this hide some trial ID	string	NOTE: in next version it will be possible to insert more ID separated with a comma id1,id2,id3
special fx	This introduces a special effect to create more engagement	string	WIP
sound	This introduces a sound (mp3) to create more engagement	string	WIP
music	This introduces a music (mp3) to create more engagement	string	WIP
execute	This is a trial ID to be executed in autoplay	string	only one trial can auto-execute
qrcollectionclear	Clear the current QRCode collection	boolean	WIP

executerandom	This is a trial ID to be randomly executed in autoplay	string	WIP NOTE: it will be possible to insert more ID separated with a comma id1,id2,id3 only one will be execute
activaterandom	This is a trial ID to be randomly activate	string	WIP NOTE: it will be possible to insert more ID separated with a comma id1,id2,id3 only one will be activate

```
{
    "moreinfo": "",
    "activate": "",
    "specialefx": "",
    "execute": ""    }
```

TRIAL AFTER FAILED

This structure describes some action after a trial is failed.

Function	Description	Format	Note
more info	This permit to add a string with suggestions, feedbacks, engaging messages	string	
activate	This activates the following next trial ID to activate in the list of trials	string	NOTE: in next version it will be possible to insert more ID separated with a comma id1,id2,id3
special fx	This introduces a special effect to create more engagement	string	WIP
sound	This introduces a sound (mp3) to create more engagement	string	WIP
music	This introduces a music (mp3) to create more engagement	string	WIP
execute	This is a trial ID to be executed in autoplay	string	only one trial can auto-execute
qrcollectionclear	Clear the current QRCode collection	boolean	WIP
score	score lose	integer	
eventfailed	if the event failed	bool	if the event failed

```
{
    "moreinfo": "",
    "activate": "",
    "specialefx": "",
    "execute": ""    }
```

TRIAL SCORE OPTIONS

This structure describes the score options in a trial.

Not to be used with the editor. It is possible only with manual coding.

Objects	Description	Format	Note
success	If the trial is completed with success	json object	
failed	If the trial is failed	json object	
unused	If the trial is set to “unused”	json object	

Function	Description	Format	Note
score	Add score to the game’s score	integer	set a negative value to subtract the score
time	Add time (minutes) available to complete the game	integer	set a negative value to subtract the time

```
{
    "success":{ “score":0,”time”:0},
    "failed":{ “score":0,”time”:0},
    "unused":{ “score":0,”time”:0}
}
```

EDITOR DEBUG MODE

(ONLY PRO VERSION)

Inside the editor are present some components to help the developer to test and debug the script:

Script Debugger (W.I.P.)

The system allows you to debug the script before it is tested in the application (**installing tests in the real application is strongly recommended**).

The trials are simulated and the order in which they are executed, you can simulate the events in case of successfully completed trials or trials failed.

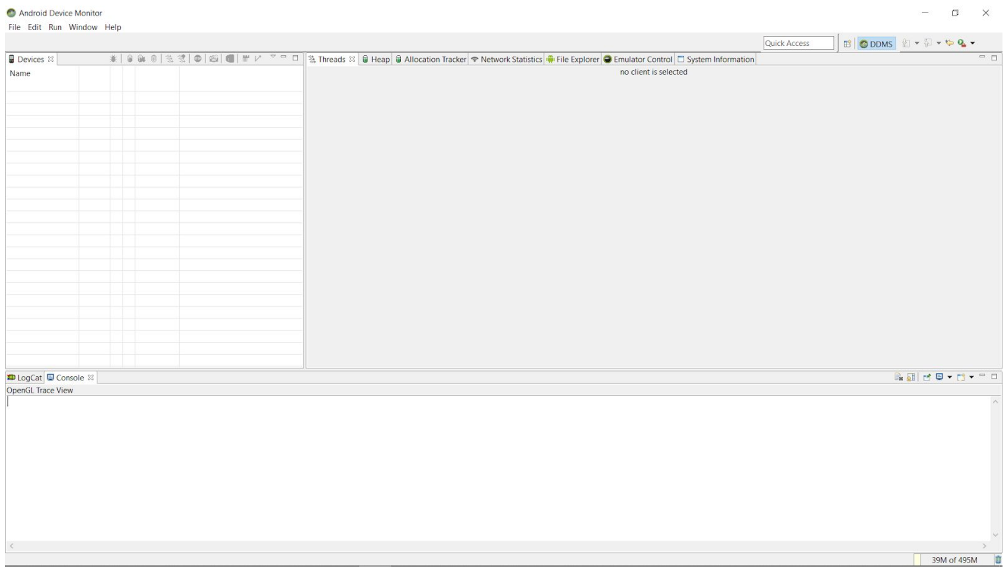
Timeline Debugger (W.I.P.)

The timeline allows you to use time management to understand how the script will behave in time trials. This system is particularly useful for testing long events, difficult to test in reality.

USB App Deploy (only Android)

You can install the application directly in the internal directory of your smartphone, to test the application before making it available to users.

Android SDK Monitor



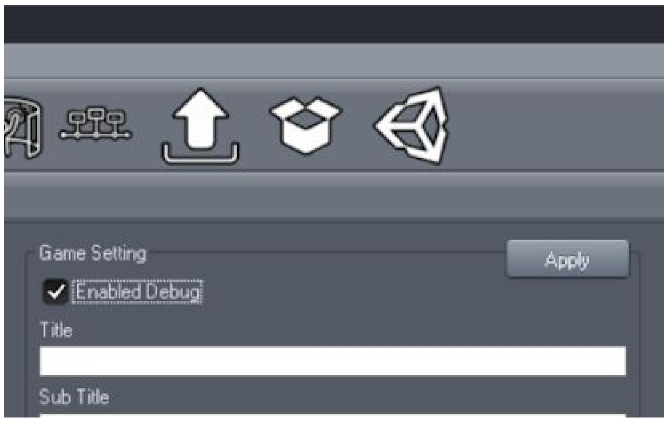
no client is selected

Using the official Android SDK you can check the functioning of the app and scripts through the Android monitor.

You must select the device on which the app is installed, select the application thread and see all the messages that the application sends to the monitor.

Script's Debug Mode

To enabled the DEBUG, please check Enabled Debug on Game Setting.

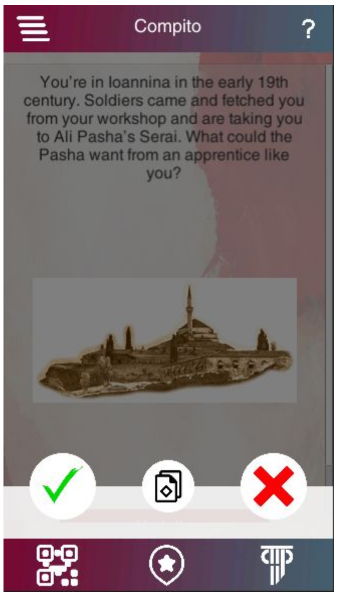


✓ Enabled Debug

Use this icon to connect with the internal debug system.

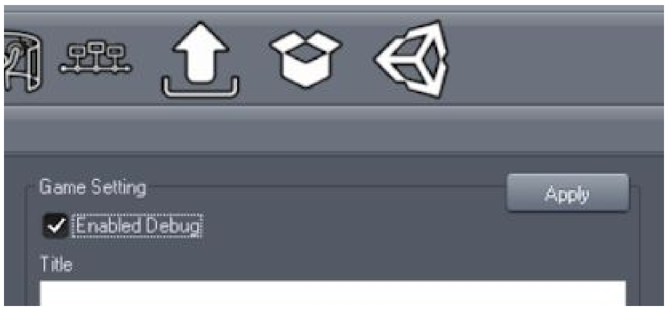
ON APP DEBUG

The application can run in debug mode to test and debug the script.
With this mode you can test scripts quickly and easily, without necessarily having to use the script in the real world.



You're in Ioannina in the early 19th century. Soldiers came and fetched you from your workshop and are taking you to Ali Pasha's Serai. What could the Pasha want from an apprentice like you?

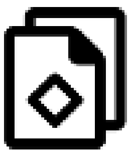
After installation you can test all the trials in debug mode (remember to activate the flags debug in the script and remove it for the official release)



Game Setting

☑ Enabled Debug

Title

		
Make a Trial "Completed"	Remove the debug mode only for the current Trial.	Make a Trial "Failed"

EDITOR

BLOCK

Each game consists of various trials, which build different blocks. They can be mixed up in several ways in order to create a story. Every block can arrange more json described in the previous sections in a visual aspect.
Here are the main blocks

Block	Description	Trial type	Json value
Link	Create a Link trial	Link	
Info	Create an Info trial	Info	
Quiz	Create a Quiz trial	Quiz	
OpenQuiz	Create an Open Quiz trial	Open Quiz	
QRCode	Create a QR Code trial	QRCode	
GPS	Create a GPS trial	GPS	
Score	Create a Score trial	Score	
Level	Create a Level trial	Level	

BRICK

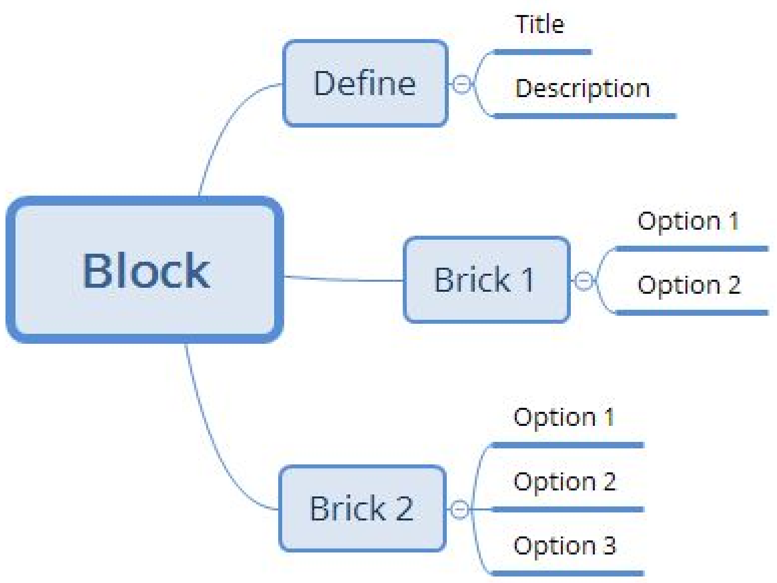
The bricks represent the components which create the blocks; the blocks activate a series of functionalities which create the game. Most of the times, each brick activates a specific functionalities on the trials. For this reason, **the main suggestion is not to fill too many bricks inside the block.**

Brick	Description	Options	BLOCKS Accepting
Define	Create a definition of the BLOCK		All
Text	Create the text options		Info

Quiz	Create the quiz options		Quiz
Open Quiz	Create the open quiz options		Open Quiz
QRCode	Create the QRCode options		QRCode
Score	Create the score’s options		
Level	Create the level’s options		Only on new standalone game
Timer	It activates a wait for a given amount of seconds		
Click	It activates a suspension, waiting for a click		

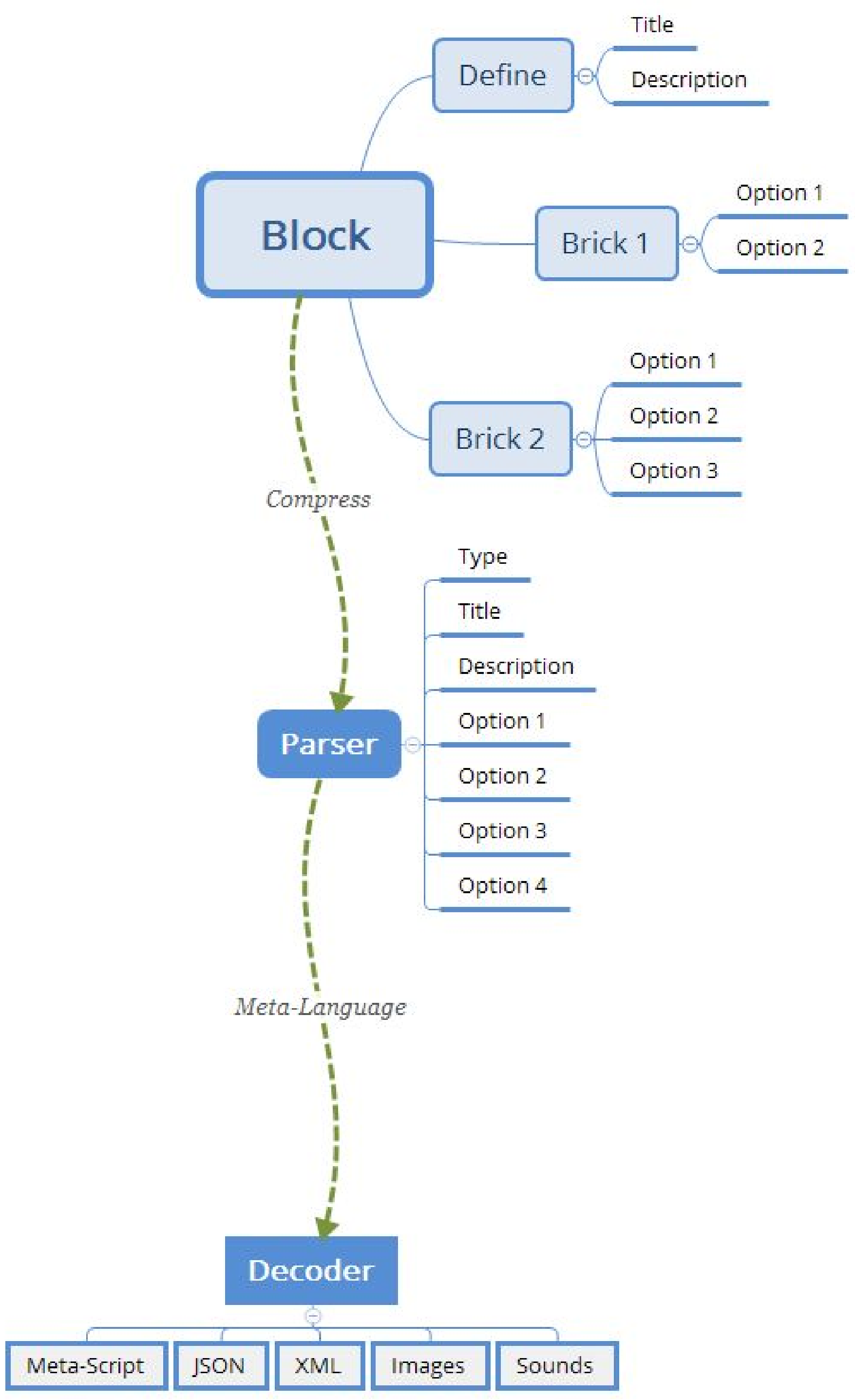
BLOCK & BRICK PHILOSOPHY

Developing applications is very complicated. This “Block & Brick” system allows you to develop applications in a simple way, as it will be the parser to decode in different languages / scripts.



Each block can have different bricks that determine its functionality.

The block determines the main category but it is the bricks that determine how the block is executed, so the same blocks, with different bricks, have different functionality.

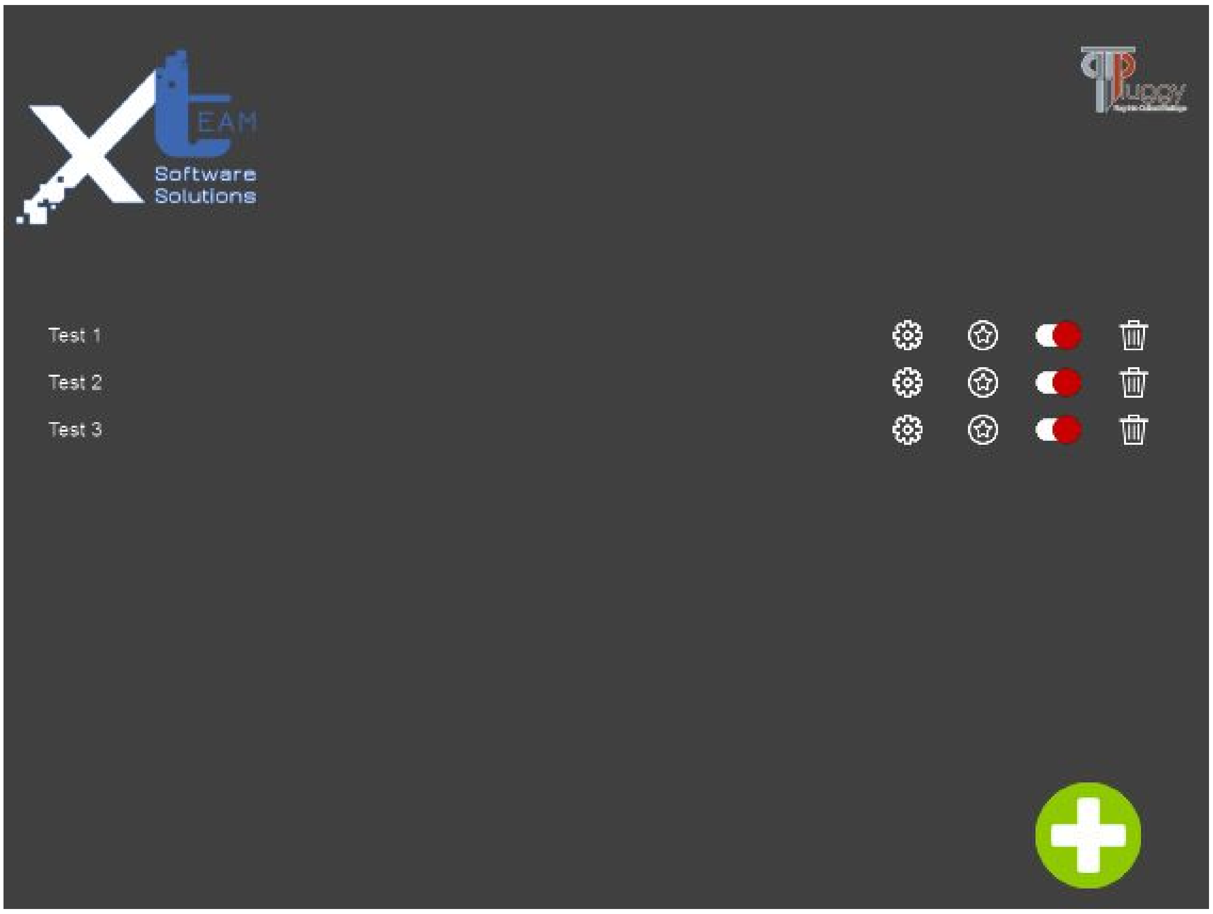


The decoding process is structured in 3 steps:

1. The visual structures composed of Block & Brick are compressed by a PARSER that creates a unified structure.
2. PARSER creates an intermediate language (meta-language) independent of the target platform and the type of script.
3. the meta-language is decoded by a DECODER that creates the different types of files needed to run the script on the target devices.

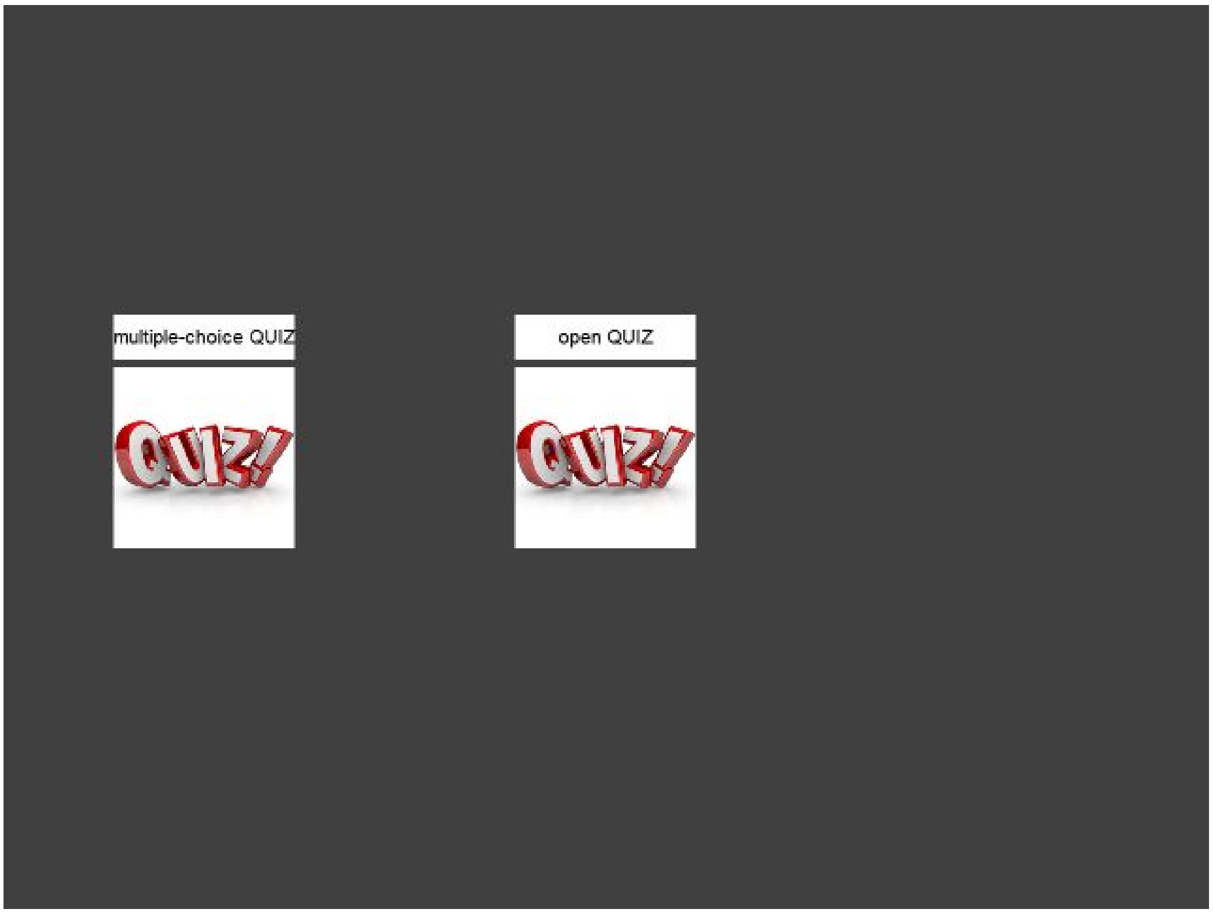
USE THE WEB EDITOR

Select a script



NOTE: to prevent overloading of scripts in the platform. The maximum number of scripts per user has been set to 3.

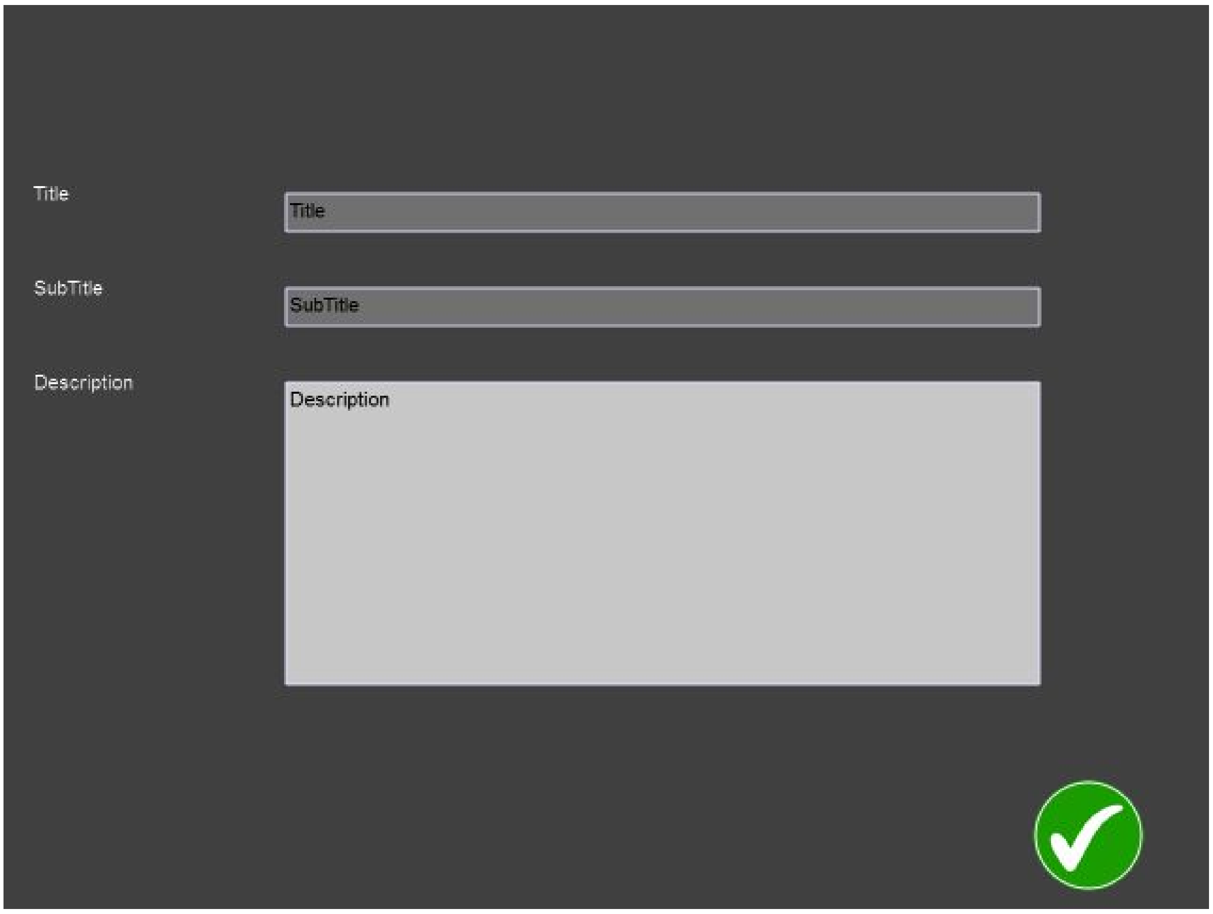
Select a template



Templates are essentially used to simplify script development work.

Templates are created for the user who will develop the game scripts, they should be as simple as possible

Script Setting

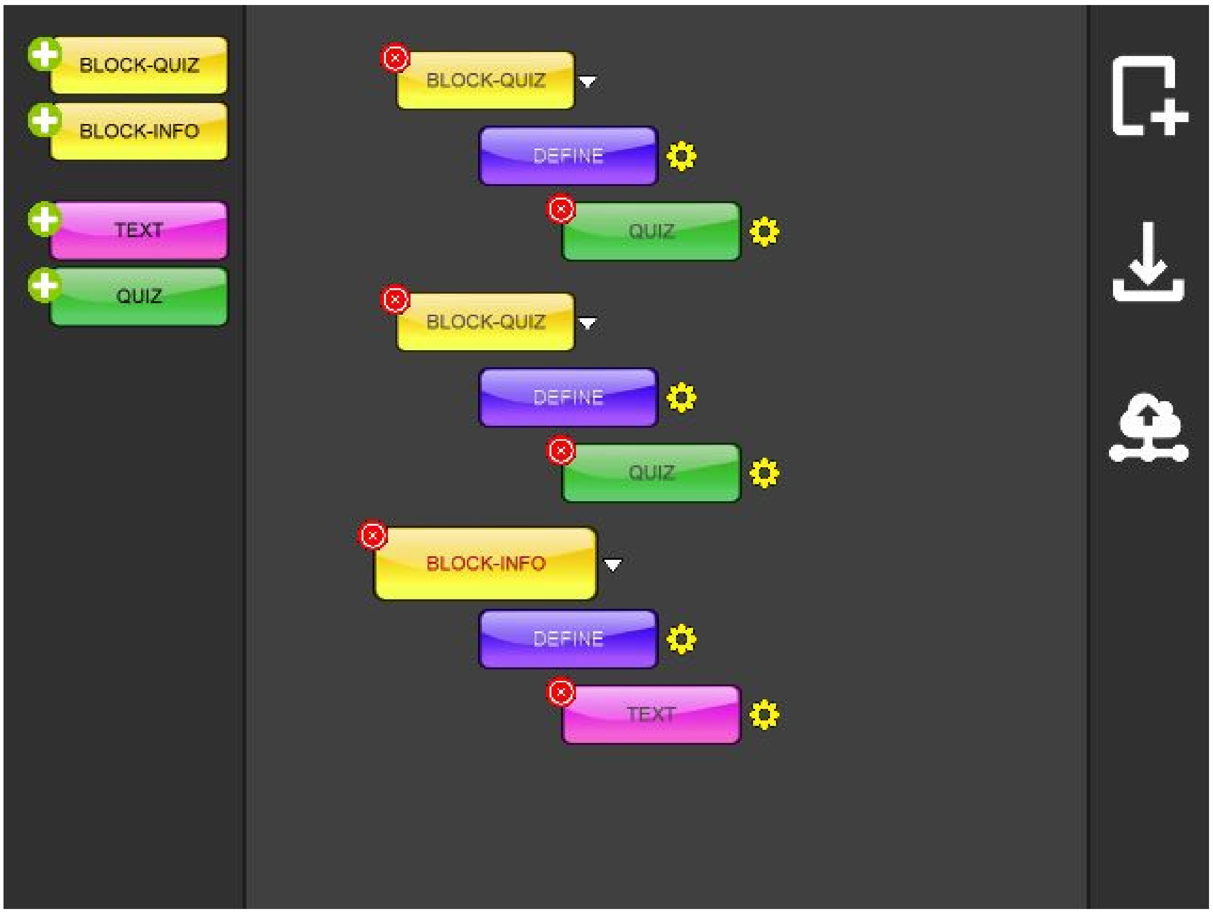


This section can be used to enter the basic parameters of the script:

- Title
- SubTitle
- Description

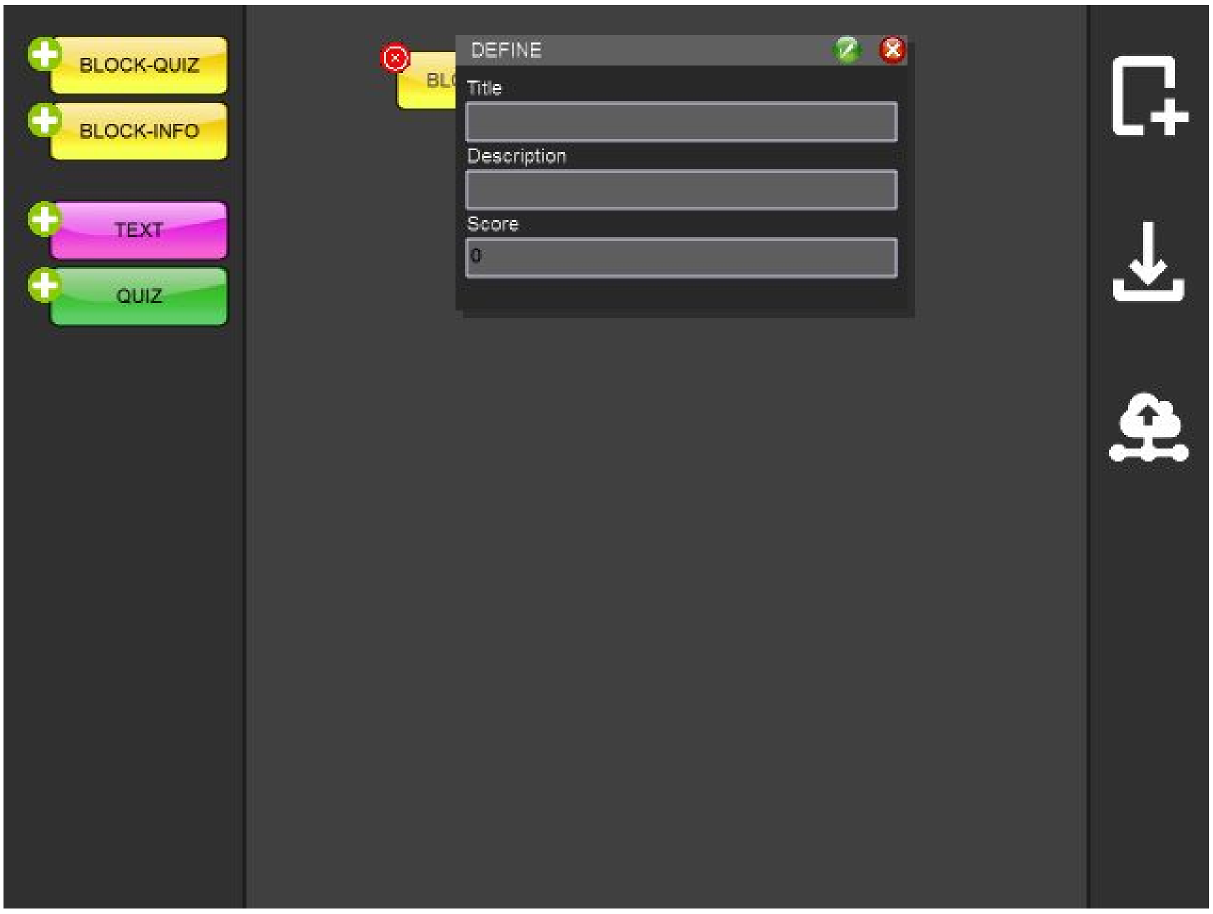
The uuid are generated automatically

Create the trials



Clear the current project	Save the current project	Upload the current project to PLUGGY

Define a trial



Title

Description

Score

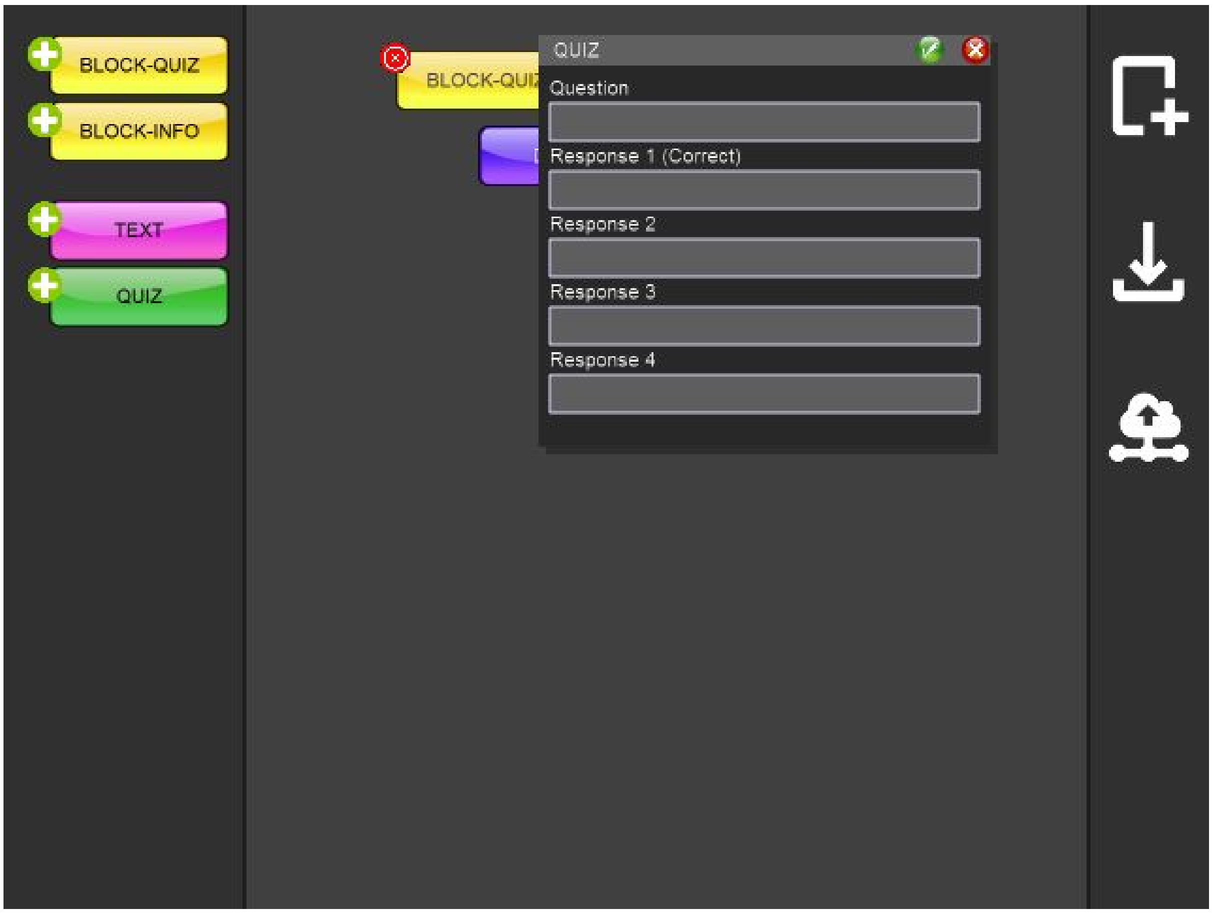
0

The minimum definition of a block:

- title
- description
- score

You must remember that the first block is activated by default, the other blocks are activated in a consequential way and hide the old blocks.
The aftercomplete and afterfailed structures are automatically created.

Setting a Brick



QUIZ

Question

Response 1 (Correct)

Response 2

Response 3

Response 4

Each brick has parameters that determine the operation of the blocks.
The parameters displayed are the minimum parameters of use.

Brick parameters may be different in different blocks. This structure is necessary in order to configure the trials as easily as possible.

ADDED A TEMPLATE

It is possible to develop additional templates, the configuration files are xml files

In the directory

data/resources/

the different templates are inserted. The configuration of template is very simple, you can insert resources directly into the template and disabled/enabled block and brick.

By default blocks and bricks are disabled, so the template must enabled the blocks/bricks to be inserted.

```
 <enabled>

        <item name="BLOCK-QUIZ" value="1" />
        <item name="BLOCK-INFO" value="1" />

        <item name="TEXT" value="1" />
        <item name="QUIZ" value="1" />
</enabled>
```

In a quiz template, it's useful to activate only a few blocks and bricks to make development easy.

Added asset

```
<background>

        <item name="background" class="IMAGE" value="1" path="backgrounds/001.jpg" id="0" width="640"
height="1024" accept="jpg"
                resource0="backgrounds/001.jpg"
                resource1="backgrounds/002.jpg"
                resource2="backgrounds/003.jpg"
        />
  </background>
  <sprite name="asset1" >

        <item name="sprite 1" class="IMAGE" value="1" path="fruits/000.png" id="0" width="128" height="128"
accept="png"
                resource0="fruits/000.png"
                resource1="fruits/001.png"
                resource2="fruits/002.png"
                resource3="fruits/003.png"
                resource4="fruits/004.png"
                resource5="fruits/005.png"
                resource6="fruits/006.png"
                resource7="fruits/007.png"
```

```
                        resource8="fruits/008.png"
                        resource9="fruits/009.png"
                        resource10="fruits/010.png"
                        resource11="fruits/011.png"
                        resource12="fruits/012.png"
                        resource13="fruits/013.png"
                        resource14="fruits/014.png"
                        resource15="fruits/015.png"
            />
    </sprite>
    <music>
            <item name="music" class="MUSIC" value="1" path="musics/001.mp3" id="0"
                        resource0="musics/001.mp3"
                        resource1="musics/002.mp3"
                        resource2="musics/003.mp3"
                        resource3="musics/004.mp3"
            />
    </music>
    <sfx>
            <item name="sfx" class="SFX" value="1" path="sfx/001.mp3" id="0"
                       resource0="sfx/001.mp3"
                        resource1="sfx/002.mp3"
                        resource2="sfx/003.mp3"
                        resource3="sfx/004.mp3"
            />

    </sfx>
```

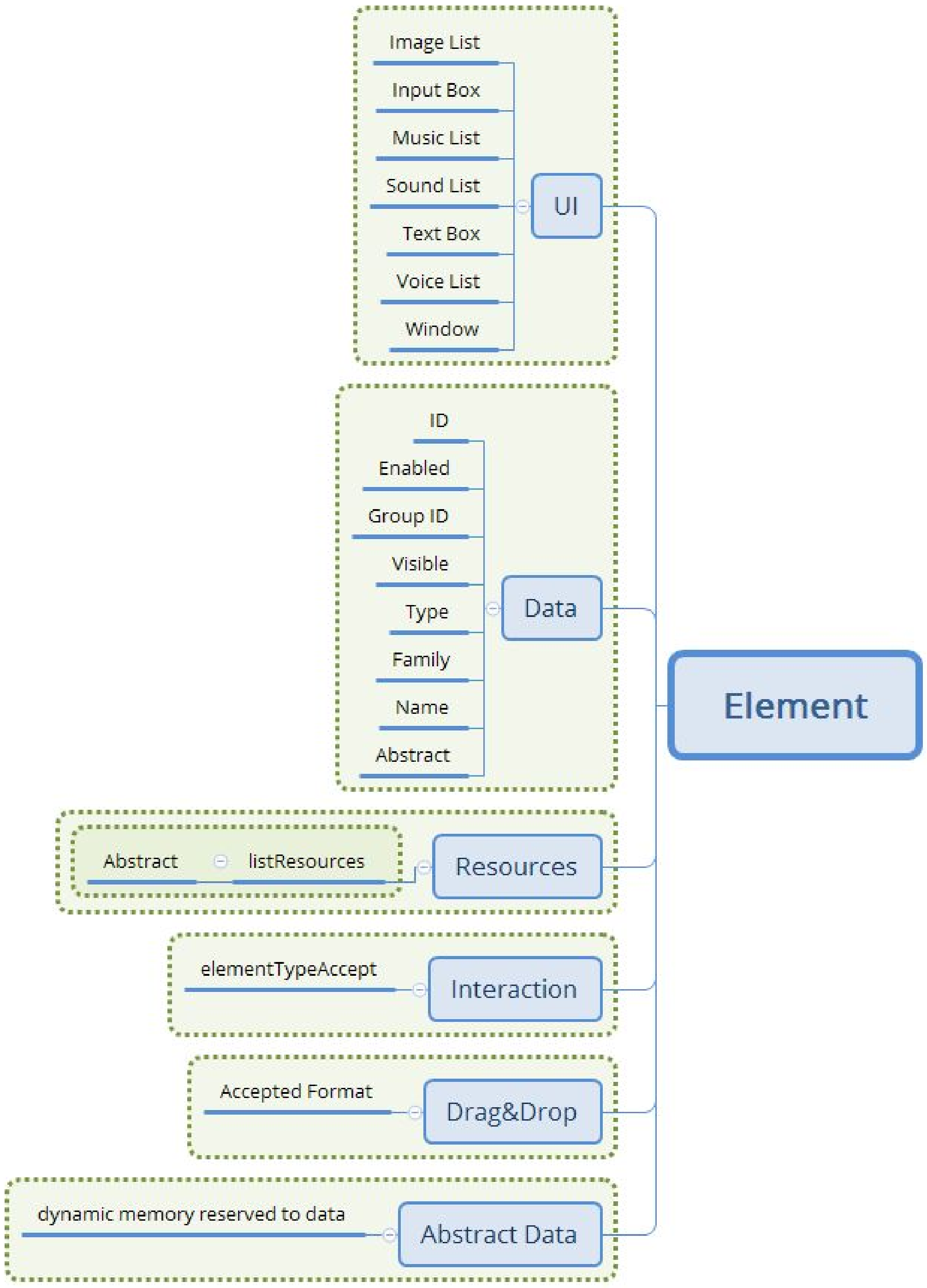
Is possible also add default resources (assets):

- background (jpeg/png)
- sprite (jpeg/png)
- music (mp3)
- sfx (mp3)

ELEMENT DEFINITION

Each element (block or brick) is composed of a dynamic data structure, composed of some basic parameters automatically created:

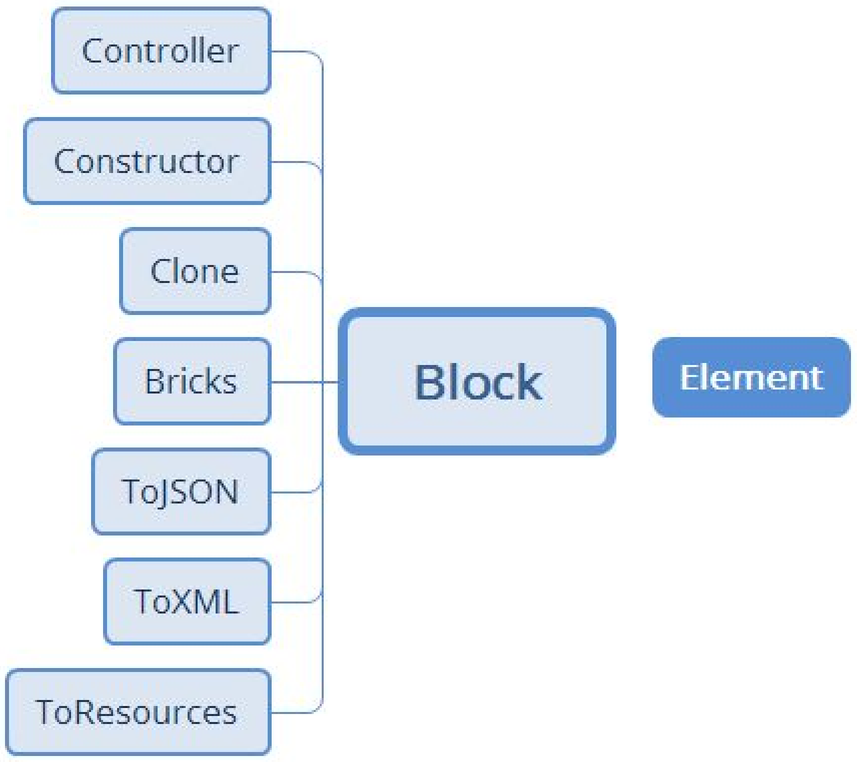
Dynamic memory zones are allocated in relation to the types of data associated with the element. Multiple elements with different dynamic memory zones can be developed



CREATE A NEW BLOCK

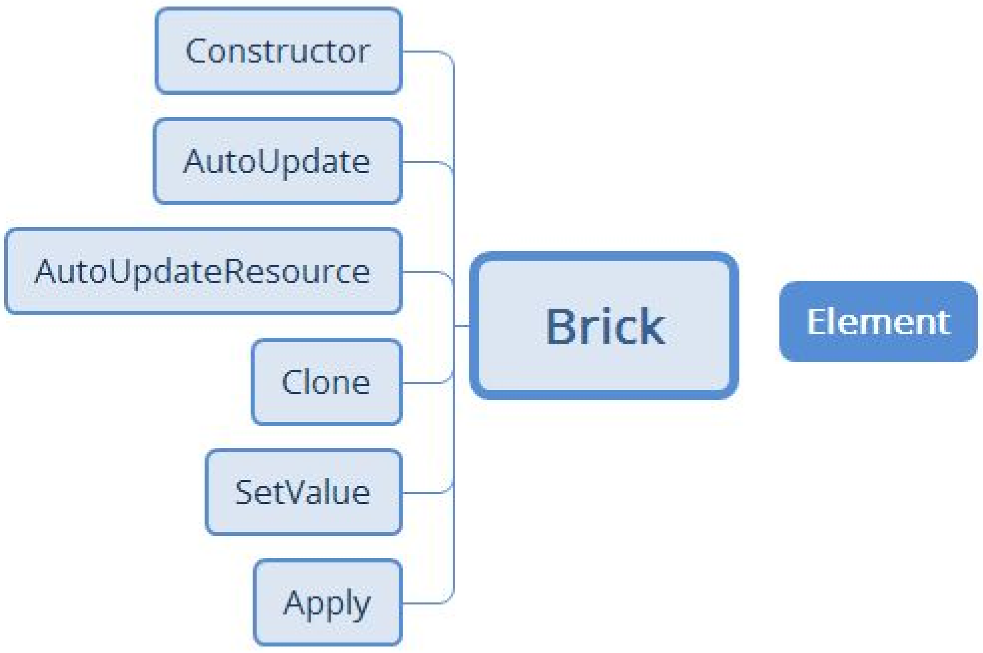
A block is the highest level of trial definition, it is the element that controls the trial.

Data structures are generated and controlled automatically



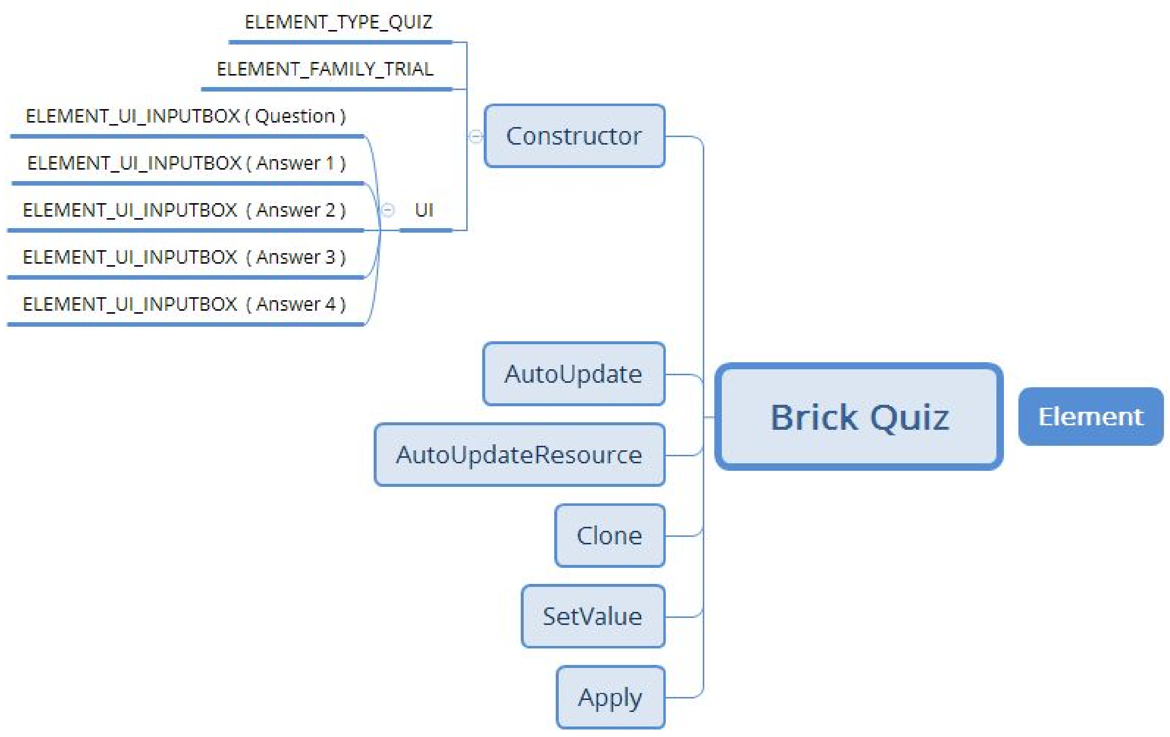
CREATE A NEW BRICK

Brick are the distinctive elements that make up the trials, allowing you to create always different trials.



BRICK DEFINITION EXAMPLE

Let's demonstrate an example of brick development, let's look at the quiz.


ELEMENT_TYPE_QUIZ
ELEMENT_FAMILY_TRIAL
ELEMENT_UI_INPUTBOX (Question)
ELEMENT_UI_INPUTBOX (Answer 1)
ELEMENT_UI_INPUTBOX (Answer 2)
ELEMENT_UI_INPUTBOX (Answer 3)
ELEMENT_UI_INPUTBOX (Answer 4)
UI
Constructor
AutoUpdate
AutoUpdateResource
Clone
SetValue
Apply
Brick Quiz
Element

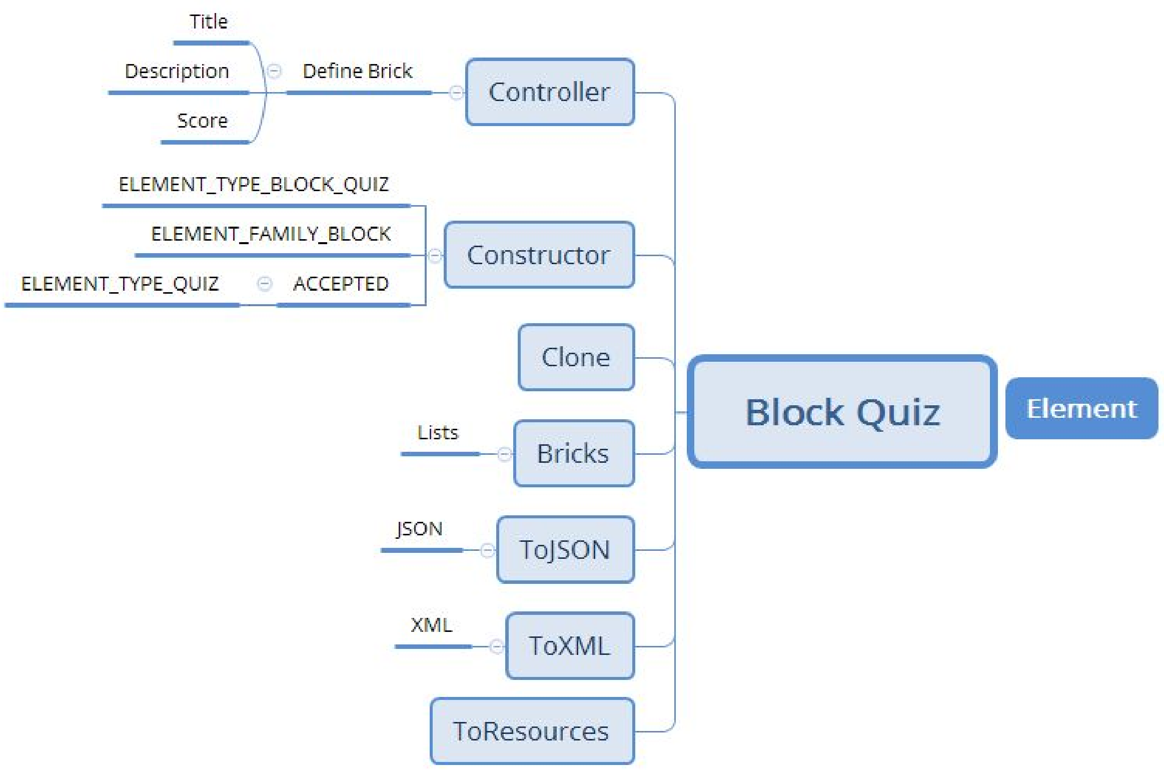
It is necessary to add the constructors who define the behavior of the brick.
All brick data structures will be managed automatically.

the data will be automatically managed using the SetValue and Apply functionalities.

To add a new answer, simply need to add ELEMENT_UI_INPUTBOX in the Constructor.

Block Definition Example

Let's demonstrate an example of block development, let's look at the quiz.



The functionality of the block is influenced by the bricks that can be hooked (and possibly also by their position in the list of bricks)

For example that we modify the constructor with the possibility to accept brick image:

ELEMENT_TYPE_IMAGE ACCEPTED

If we add a quiz brick to the quiz block, the resulting data structure :

```
{
  "id": 1,
  "type": "quiz",
  "name": "name",
  "description": "description",
  "score": 50,
  "options": {
            "quiz": {
            "question": "question",
            "answer": {
            "1",
            "2",
            "3",
            "4"
            },
            "casesensitive": false
            }
  }
}
```

If we add also an image brick to the quiz block, the resulting data structure :

```
{
  "id": 1,
  "type": "quiz",
  "name": "name",
  "description": "description",
  "score": 50,
  "options": {
            "quiz": {
            "photo": "url",
            "question": "question",
            "answer": {
            "1",
            "2",
            "3",
            "4"
            },
            "casesensitive": false
            }
  }
}
```

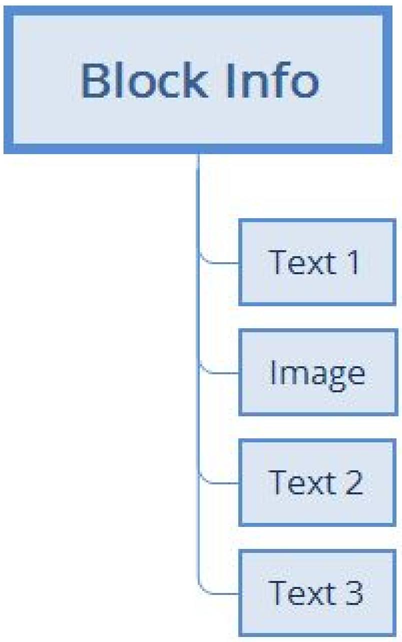
During saving, the resources present (or selected) in the image brick will be automatically created/saved.

Abstract data management and Polymorphic code

All Block & Brick management code is managed by an "undefined" data structure manager, this "undefined" manager is composed of as many other "undefined" managers as the number of blocks and bricks added in the project.

This management system allows you to change Block & Brick data structures in real time, changing resources / structures management without modifying the original data structure .

Warning! These examples are for teaching purposes only. It has been disabled in the editor in order to avoid confusion in the user.



Let's assume you create an **Info Block** and add a **Brick Text** , the resulting data structure will be a trial info where the description is contained in the text block.

```
"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
"description" = BRICK TEXT -> Text
```

Now let's add an **Image Brick**: the options management will be automatically created with the link to the image

```
"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
“description" = BRICK TEXT 1 -> Text
“options”
        “photo_url” =     BRICK IMAGE -> photo
```

Now we add an additional **Text Brick**. There are no more descriptions field or infos free and an “aftercomplete” structure with the caption “more info” is automatically created.

```
"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
“description" = BRICK TEXT 1 -> Text
“options”
        “photo_url” =       BRICK IMAGE -> photo
“aftercomplete”
        “more_info”=        BRICK TEXT 2 -> Text
```

Now we add an additional **Text Brick**: the Info Blocks don’t have an “afterfailed” structure (the trial info can't fail, it can just be completed), therefore, actually no further descriptions or information can be added.

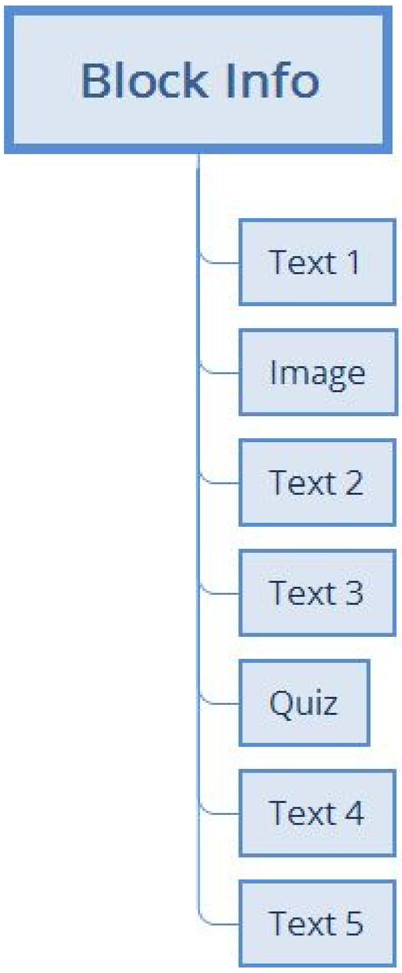
The code is automatic changed, creating a clone of the parent Info Block (included the definition). In this case we get a data structure:

```
"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
“description" = BRICK TEXT 1 -> Text
“options”
        “photo_url” =       BRICK IMAGE -> photo
“aftercomplete”
        “more_info”=        BRICK TEXT 2 -> Text


"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
“description" = BRICK TEXT 3 -> Text
```

Up to this point it is quite easy to understand how the "undefined" manager works.

Now we complicate it by inserting a brick that is completely different from the existing ones: a **Quiz Block.** In this case a trial quiz will be created, the data structure is filled with the parameters used during the definition of the quiz



```
"type" = "info"
"name" = BLOCK  INFO -> Define -> Title
"description" = BRICK TEXT 1 -> Text
"options"
        "photo_url" =     BRICK IMAGE -> photo
"aftercomplete"
        "more_info"=      BRICK TEXT 2 -> Text


"type" = "info"
"name" = BLOCK  INFO -> Define -> Title
"description" = BRICK TEXT 3 -> Text

"type" = "quiz"
"name" = BLOCK  INFO -> Define -> Title
"description" = BLOCK  INFO ->  Define -> Desc
"options"
        "quiz"
                "question" = BRICK QUIZ -> question
                "answer1" = BRICK QUIZ -> answer 1
                "answer2" = BRICK QUIZ -> answer 2
                "answer3" = BRICK QUIZ -> answer 3
                "answer4" = BRICK QUIZ -> answer 4
```

Now let's add another **Text Brick**, the manager creates an “aftercompleted” data structure.

```
"type" = "info"
```

```
"name" = BLOCK  INFO -> Define -> Title
"description" = BRICK TEXT 1 -> Text
"options"
        "photo_url" =       BRICK IMAGE -> photo
"aftercomplete"
        "more_info"=        BRICK TEXT 2 -> Text


"type" = "info"
"name" = BLOCK  INFO -> Define -> Title
"description" = BRICK TEXT 3 -> Text

"type" = "quiz"
"name" = BLOCK  INFO -> Define -> Title
"description" = BLOCK  INFO ->  Define -> Desc
"options"
        "quiz"
                "question" = BRICK QUIZ -> question
                "answer1" = BRICK QUIZ -> answer 1
                "answer2" = BRICK QUIZ -> answer 2
                "answer3" = BRICK QUIZ -> answer 3
                "answer4" = BRICK QUIZ -> answer 4
"aftercomplete"
        "more_info"=        BRICK TEXT 4 -> Text
```

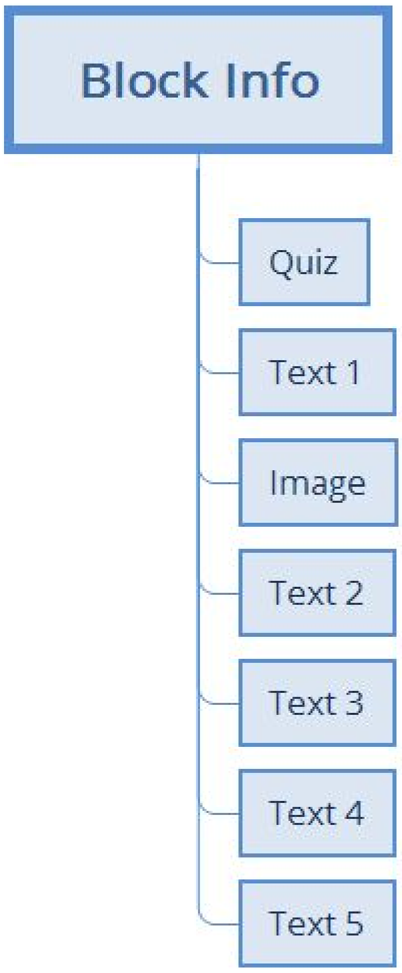
If we add a new **Text Brick**, an "afterfailed" structure will be created (in this case a trial quiz may actually fail)

```
"type" = "info"
"name" = BLOCK  INFO -> Define -> Title
"description" = BRICK TEXT 1 -> Text
"options"
        "photo_url" =       BRICK IMAGE -> image
"aftercomplete"
        "more_info"=        BRICK TEXT 2 -> Text


"type" = "info"
"name" = BLOCK  INFO -> Define -> Title
"description" = BRICK TEXT 3 -> Text

"type" = "quiz"
"name" = BLOCK  INFO -> Define -> Title
"description" = BLOCK  INFO ->  Define -> Desc
"options"
        "quiz"
                "question" = BRICK QUIZ -> question
                "answer1" = BRICK QUIZ -> answer 1
                "answer2" = BRICK QUIZ -> answer 2
                "answer3" = BRICK QUIZ -> answer 3
                "answer4" = BRICK QUIZ -> answer 4
"aftercomplete"
        "more_info"=        BRICK TEXT 4 -> Text
"afterfailed"
        "more_info"=        BRICK TEXT 5 -> Text
```

Now we move one of the bricks: for example we move the **Quiz Brick** to the beginning of the list. How will the "undefined" manager change the data structure?



The info block has no longer additional values, so it will only consist of the definition. The **Quiz Brick** creates a trial quiz, the **Text Brick 1** and the **Image Brick create** a new “aftercomplete” data structure in the trial quiz , the **Text Brick 2** creates a new the “afterfailed” data structure in the trial quiz.

The **Text Brick 3** automatic create a new clones of the original **Info Block** and creates a new info trial, the **Text Brick 4** will be created a new “aftercomplete” data structure in the info trial.

The **Text Brick 5** automatic create a new clones the original **Info Block** and creates a new info trial.

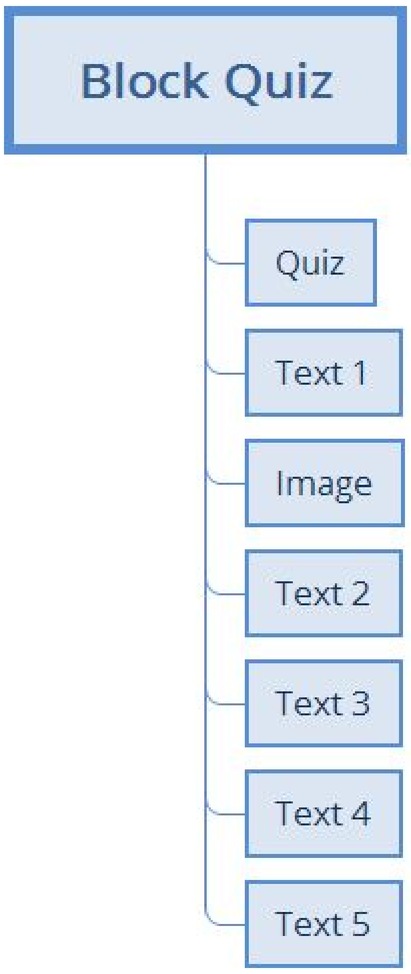
```
"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
“description" = BLOCK  INFO -> Define -> Desc

“type" = “quiz”
"name" = BLOCK  INFO -> Define -> Title
“description" = BLOCK  INFO ->  Define -> Desc
“options”
        “quiz”
```

```
                    “question” = BRICK QUIZ -> question
                    “answer1” = BRICK QUIZ -> answer 1
                    “answer2” = BRICK QUIZ -> answer 2
                    “answer3” = BRICK QUIZ -> answer 3
                    “answer4” = BRICK QUIZ -> answer 4
“aftercomplete”
          “more_info”=       BRICK TEXT 1 -> Text
          “photo”     =      BRICK IMAGE -> image
“afterfailed”
          “more_info”=       BRICK TEXT 2 -> Text

"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
“description" = BRICK TEXT 3 -> Text
“aftercomplete”
          “more_info”=       BRICK TEXT 4 -> Text

"type" = “info”
"name" = BLOCK  INFO -> Define -> Title
“description" = BRICK TEXT 5 -> Text
```



Now let's change the block type (**Block Quiz**) without changing the list of inserted bricks.

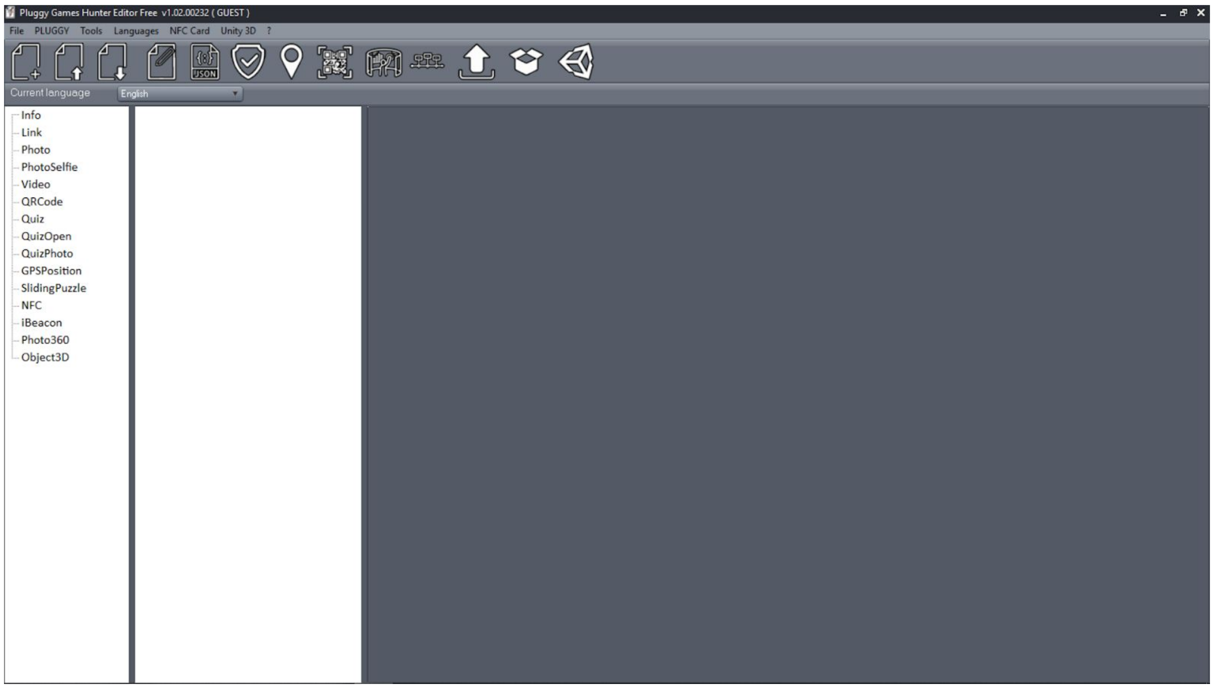
```
"type" = "quiz"
"name" = BLOCK  QUIZ -> Define -> Title
"description" = BLOCK  QUIZ ->  Define -> Desc
"options"
            "quiz"
                        "question" = BRICK QUIZ -> question
                        "answer1" = BRICK QUIZ -> answer 1
                        "answer2" = BRICK QUIZ -> answer 2
                        "answer3" = BRICK QUIZ -> answer 3
                        "answer4" = BRICK QUIZ -> answer 4
"aftercomplete"
            "more_info"=        BRICK TEXT 1 -> Text
            "photo"     =       BRICK IMAGE -> image
"afterfailed"
            "more_info"=        BRICK TEXT 2 -> Text


"type" = "quiz"
"name" = BLOCK  QUIZ -> Define -> Title
"description" = BLOCK  QUIZ ->  Define -> Desc
"options"
            "quiz"
                        "question" = ""
                        "answer1" = ""
                        "answer2" = ""
                        "answer3" = ""
                        "answer4" = ""
"aftercomplete"
            "more_info"=        BRICK TEXT 3 -> Text
"afterfailed"
            "more_info"=        BRICK TEXT 4 -> Text

type" = "quiz"
"name" = BLOCK  QUIZ -> Define -> Title
"description" = BLOCK  QUIZ ->  Define -> Desc
"options"
            "quiz"
                        "question" = ""
                        "answer1" = ""
                        "answer2" = ""
                        "answer3" = ""
                        "answer4" = ""
"aftercomplete"
            "more_info"=        BRICK TEXT 5 -> Text
```

EDITOR PRO

Advanced C++ Editor for Collaborative Game Developers



ToolBar Menu

	create a new script, select a directory where the media will be inserted
	Load a script / project
	Save a script / project

	Internal script editor, allows you to edit the game directly. Warning to use with caution
	External tool to format the json script
	External tool to verify the json script
	Google Maps tool to verify the geolocalization
	External tool to create QRCode
	Script debugger (ONLY PRO VERSION)
	Time line debugger (ONLY PRO VERSION)
	Upload to Pluggy Platform
	Create a package
	Export to Unity Asset (ONLY PRO VERSION)

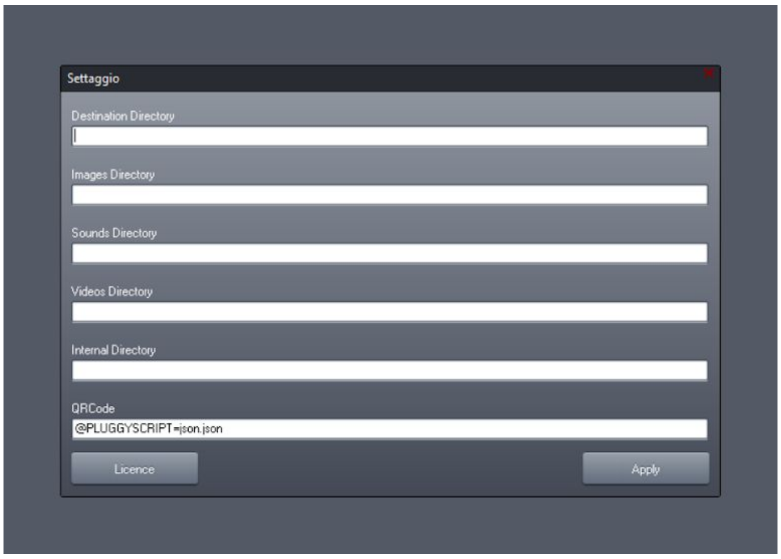
🔌	Debug system.

Main Menu

Main Menu	**Description**
File	
New	Create a new Game
Templates	Open the game’s templates
Save	Save the game script
Save as ...	Save the game script in a new folder
Load	Load a game script
Recent	List of recent projects
Close	Close the current project
Create JSON	Create the json file from the script.
Create Backup	Create a script’s backup
Parameters	Edit the game parameters
Exit	Close the editor
PLUGGY	
Login	Login in PLUGGY platform (need a PLUGGY account). Need to upload content to PLUGGY
Page	Go To PLUGGY
Logout	Log Out from PLUGGY
Tools	
Options	Set the external editor link / options
Languages	Change the editor language
NFC Cards	Use NFC Cards to create scripts (experimental)

Unity3D	
Export	Export some scripts to use with Unity asset. W.I.P.
? (Help)	
PLUGGY web site	Go to PLUGGY web site
Quick	
Info	
Support	

GAME SETTINGS


Settaggio

Destination Directory

Images Directory

Sounds Directory

Videos Directory

Internal Directory

QRCode
@PLUGGYSCRIPT=json.json

With the game setting can adapt the game to different websites, destination or target.
Destination Directory:
is possible insert a url with the destination of the script:

example
https://www.example.com/test/myscript/

Image Directory
is possible setting a directory for the images

example
https://www.example.com/test/myscript/image/

Sound Directory
is possible setting a directory for the images

example
https://www.example.com/test/myscript/snd/

Video Directory
is possible setting a directory for the videos

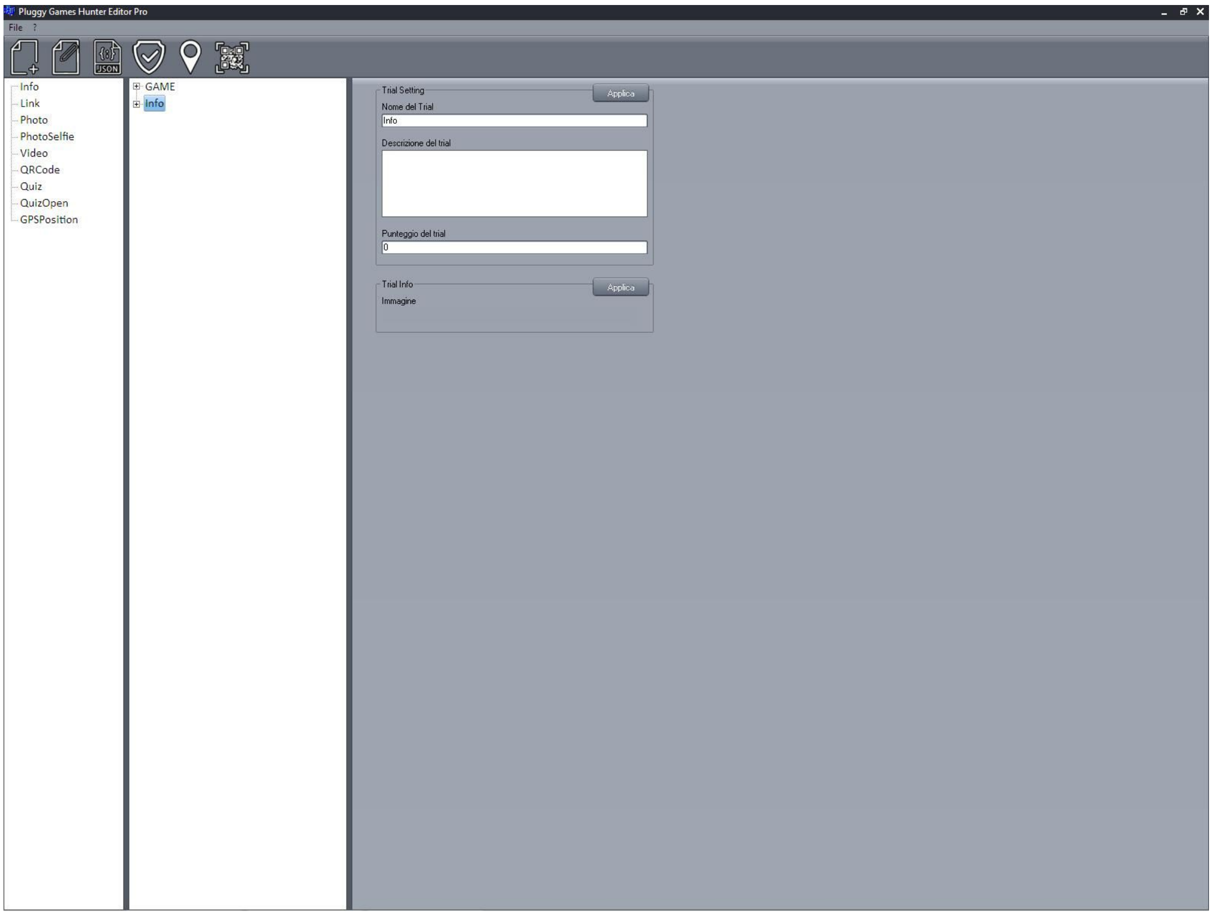
example
https://www.example.com/test/myscript/video/

Internal Directory
is possible setting a internal directory (iOs and Android) to work offline.
This directory is inside the persistent store of the application (the storage space allocated by the operating system to the application)

<persistent storage>/package/myscript

Licence button
is possible edit a custom licence for the script.

TRIAL OPTIONS



Trial Setting

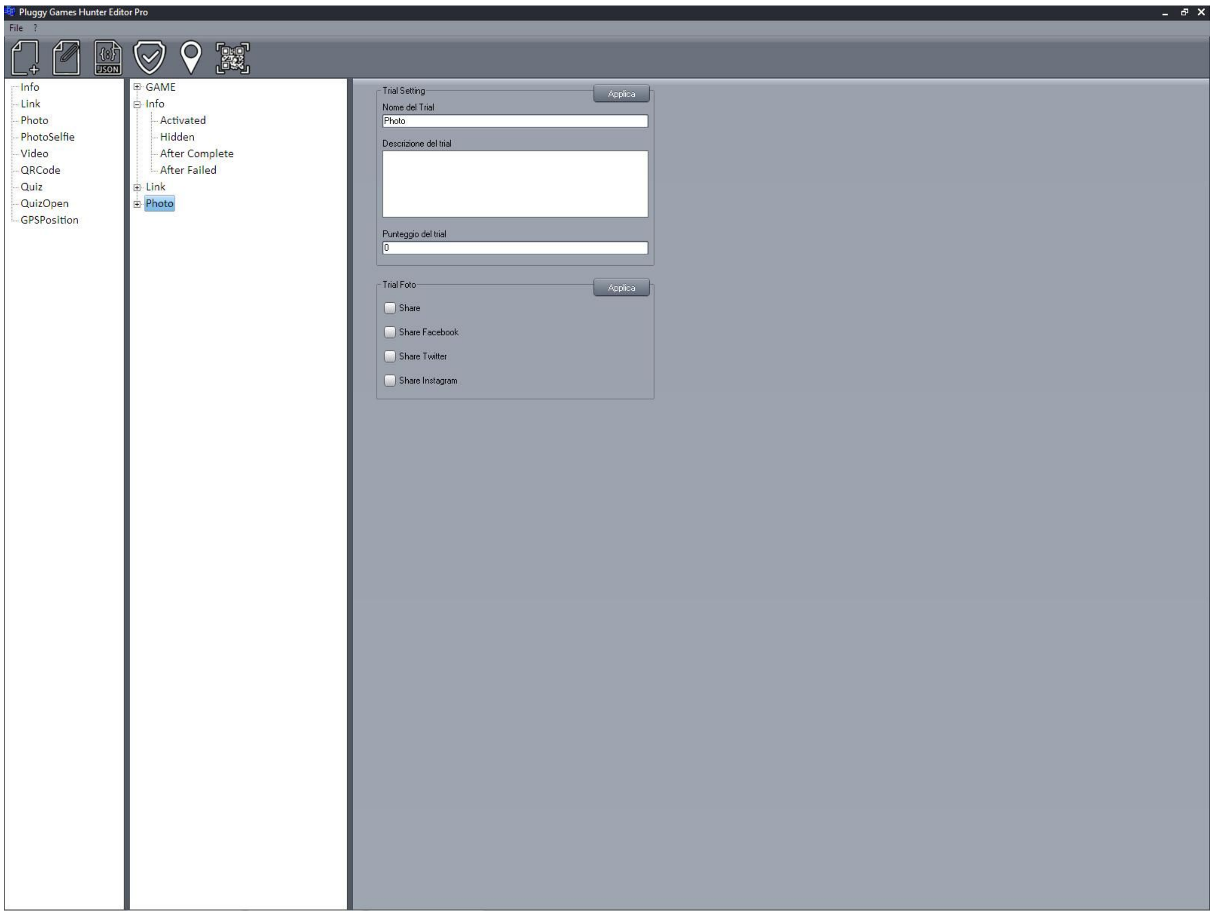
Nome del Trial: Info

Descrizione del trial

Punteggio del trial: 0

Trial Info

Immagine



Trial Setting

Nome del Trial: Photo

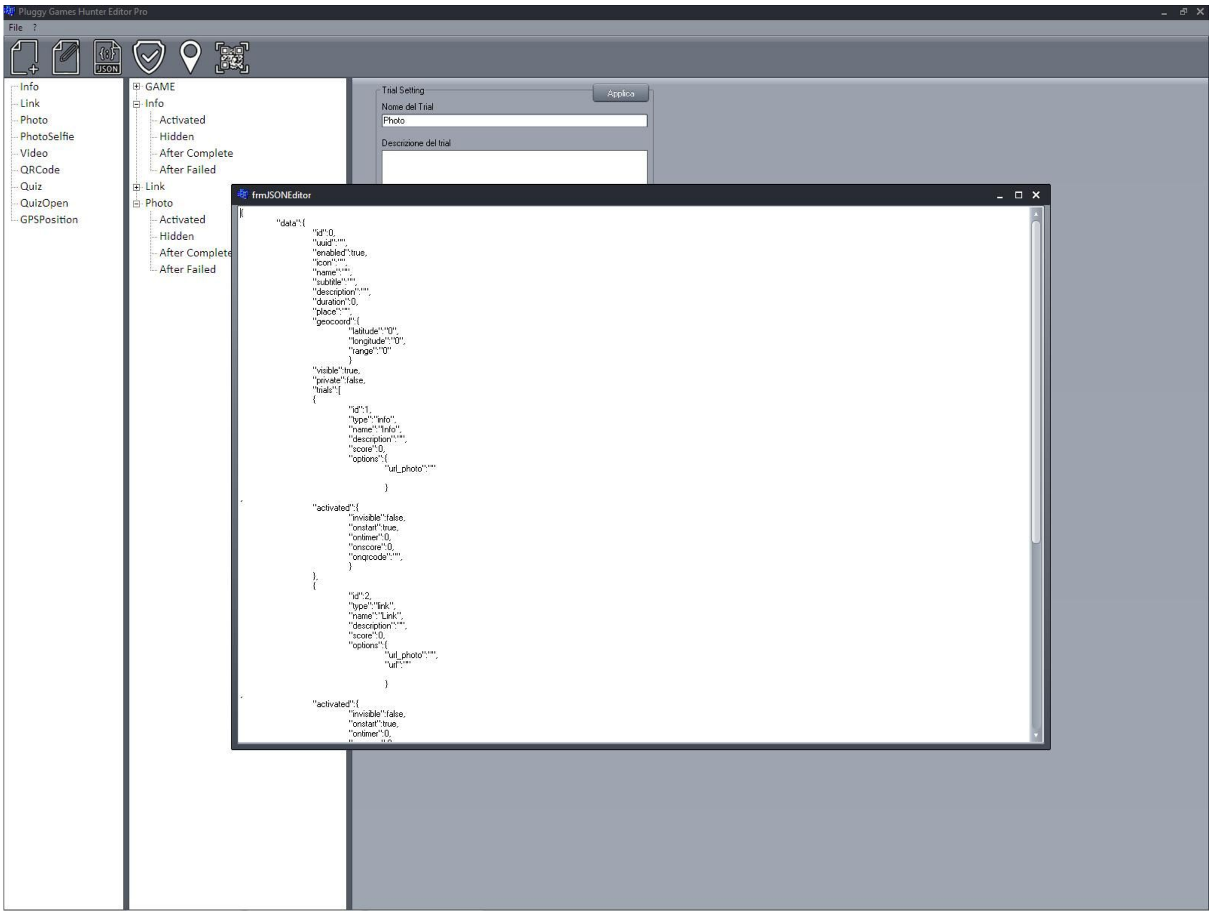
Descrizione del trial:

Punteggio del trial: 0

Trial Foto

- [ ] Share
- [ ] Share Facebook
- [ ] Share Twitter
- [ ] Share Instagram

Internal JSON Script Editor



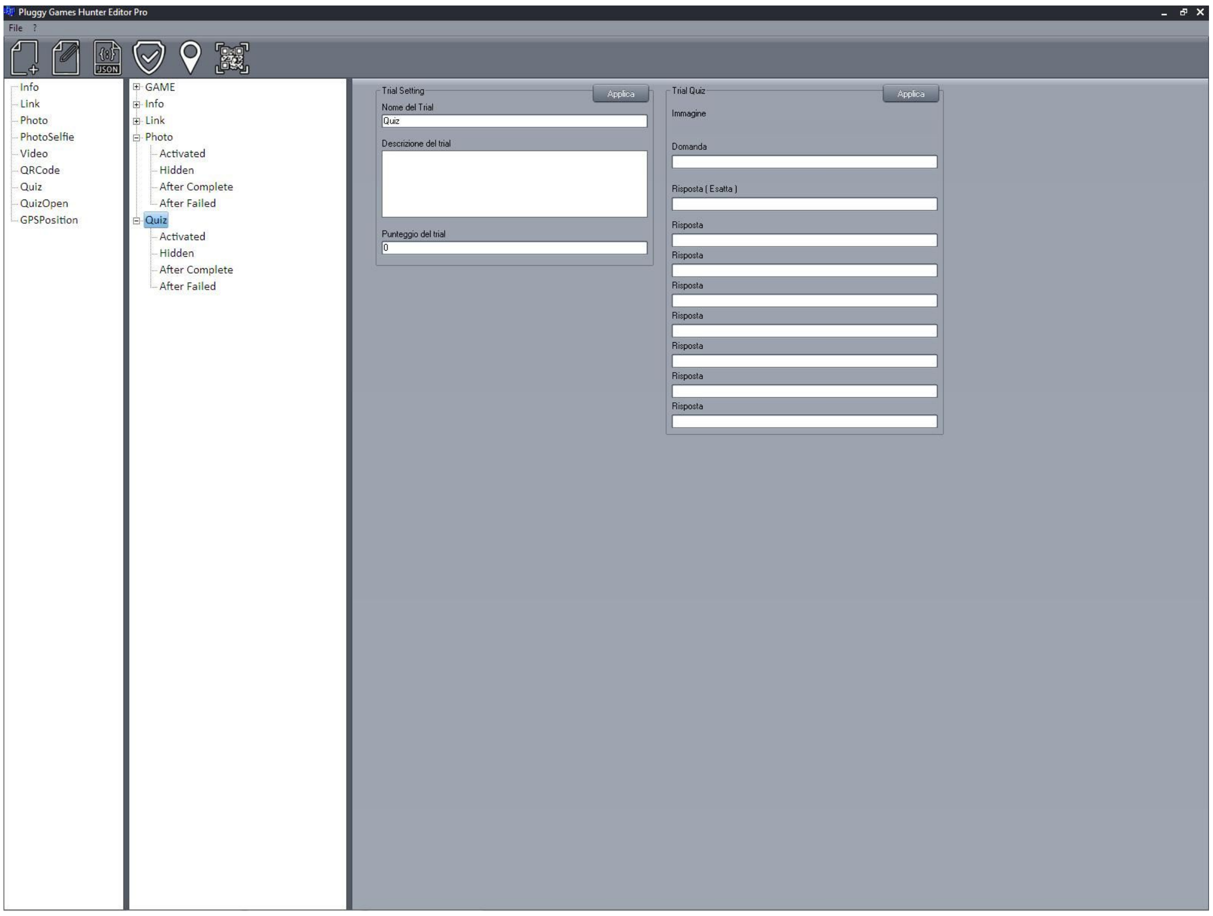
Nome del Trial: Photo

```
{
        "data":{
                "id":0,
                "uuid":"",
                "enabled":true,
                "icon":"",
                "name":"",
                "subtitle":"",
                "description":"",
                "duration":0,
                "place":"",
                "geocoord":{
                        "latitude":"0",
                        "longitude":"0",
                        "range":"0"
                        }
                "visible":true,
                "private":false,
                "trials":[
                {
                        "id":1,
                        "type":"info",
                        "name":"Info",
                        "description":"",
                        "score":0,
                        "options":{
                                "url_photo":""

                                }
,
                "activated":{
                        "invisible":false,
                        "onstart":true,
                        "ontimer":0,
                        "onscore":0,
                        "onqrcode":"",
                        }
                },
                {
                        "id":2,
                        "type":"link",
                        "name":"Link",
                        "description":"",
                        "score":0,
                        "options":{
                                "url_photo":"",
                                "url":""

                                }
,
                "activated":{
                        "invisible":false,
                        "onstart":true,
                        "ontimer":0,
```

Internal JSON Script Editor



Trial Setting

Nome del Trial: Quiz

Descrizione del trial:

Punteggio del trial: 0

Trial Quiz

Immagine

Domanda:

Risposta (Esatta):

Risposta:

Risposta:

Risposta:

Risposta:

Risposta:

Risposta:

Risposta:

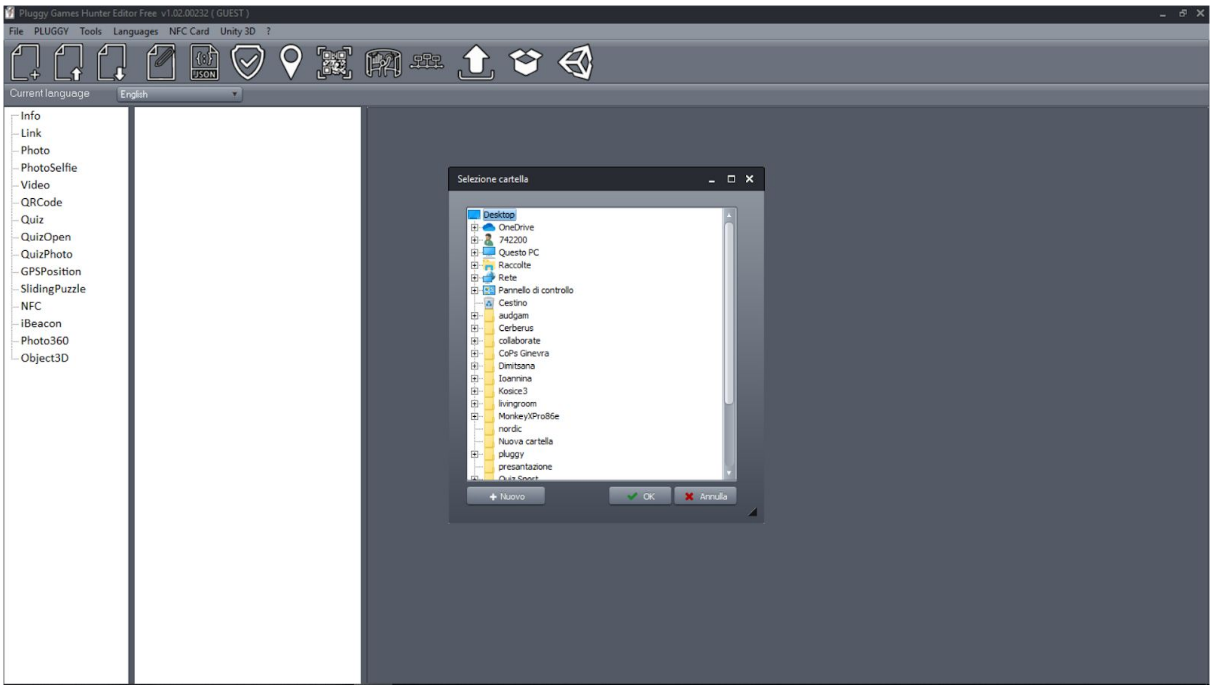
TEMPLATES REFERENCE

USE THE EDITOR PRO

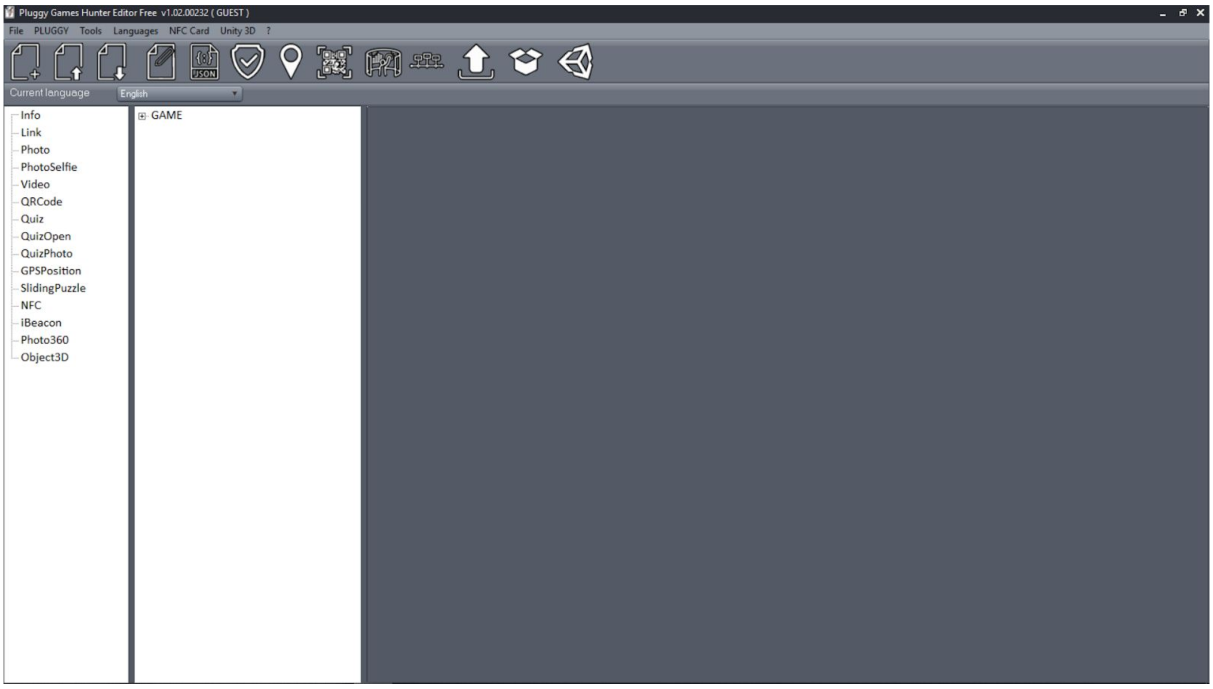
New Script

1. Click on top menu File -> New
2. Select a directory (or create a new)

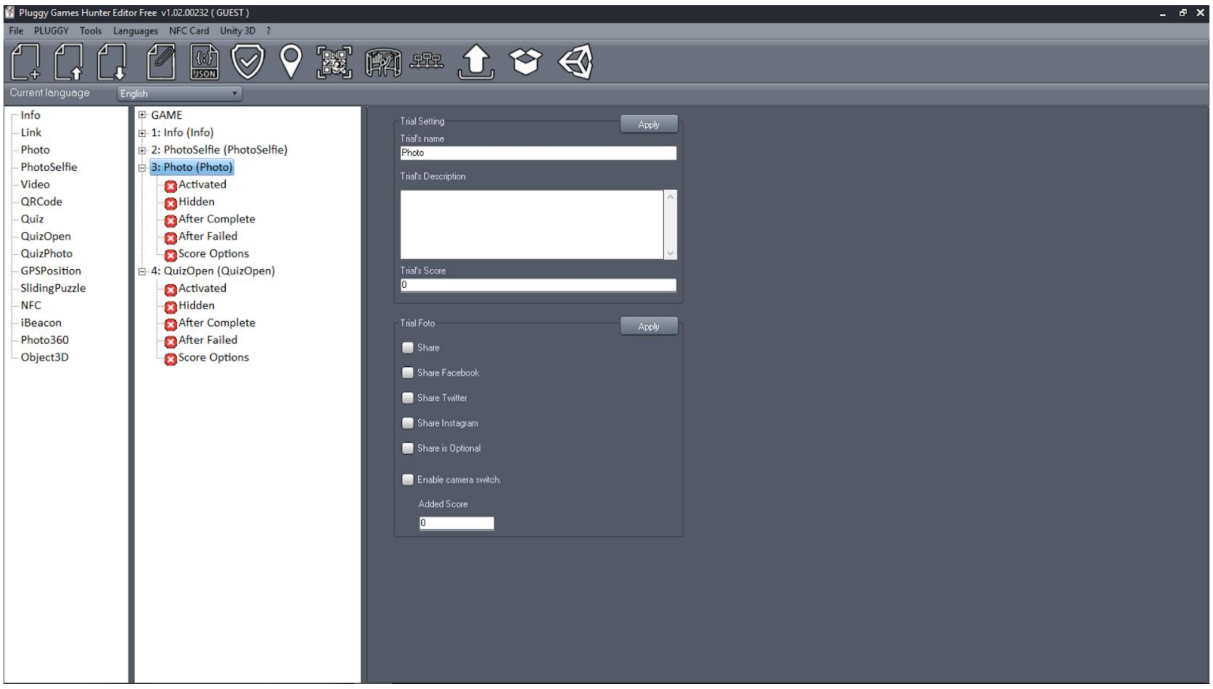
   NOTE: All the files will be stored in this directory: script, json, photos, videos, 3d objects, zip, package



3. GAME appear on the treeview list



4. Added trial from the left side tree view



Trial's name: Photo

Trial's Score: 0

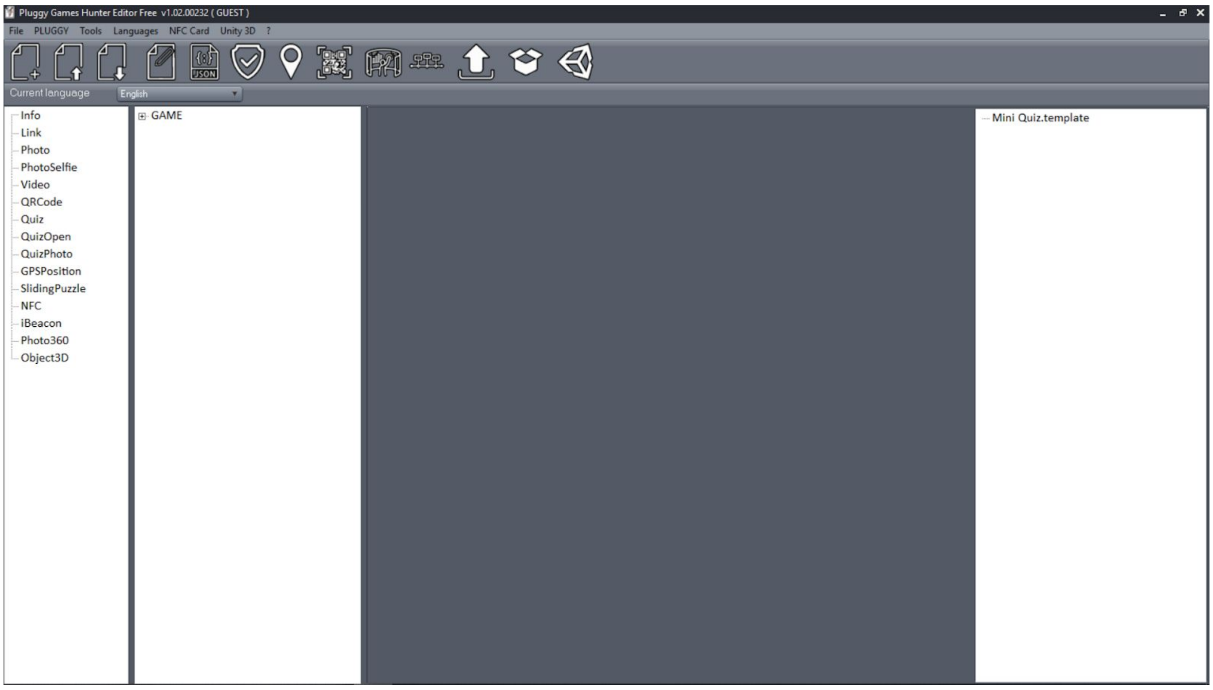
Added Score: 0

5. Click on top menu File -> Save to save the current project

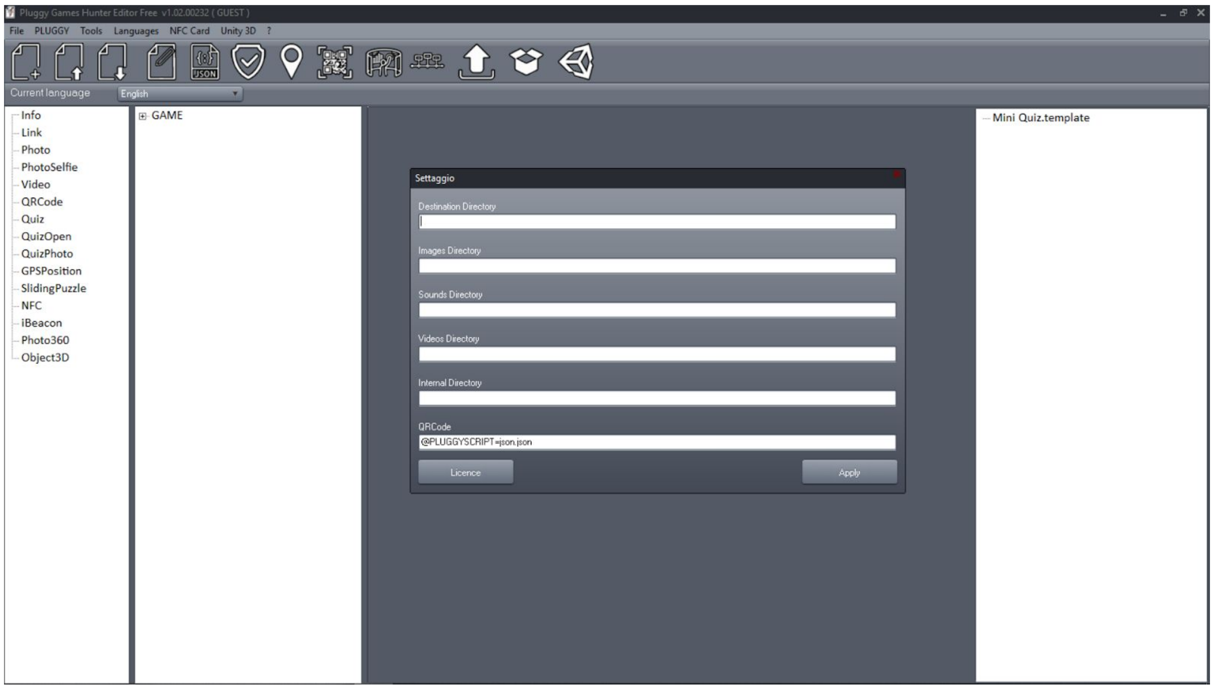
Use Templates

1. Click on top menu File -> Template
2. Select a template from the right side treeview

   NOTE: This overwrite the previous project.



Project Parameters



Settaggio

Destination Directory:

Images Directory:

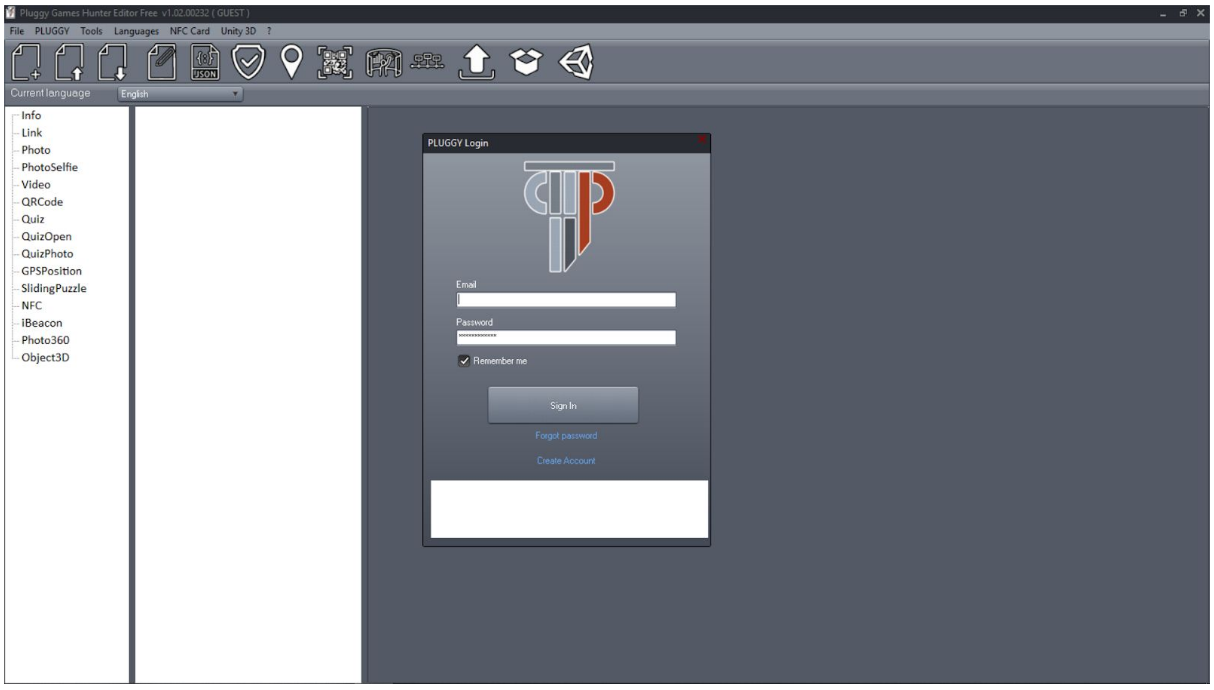
Sounds Directory:

Videos Directory:

Internal Directory:

QRCode: @PLUGGYSCRIPT=json.json

Login to PLUGGY



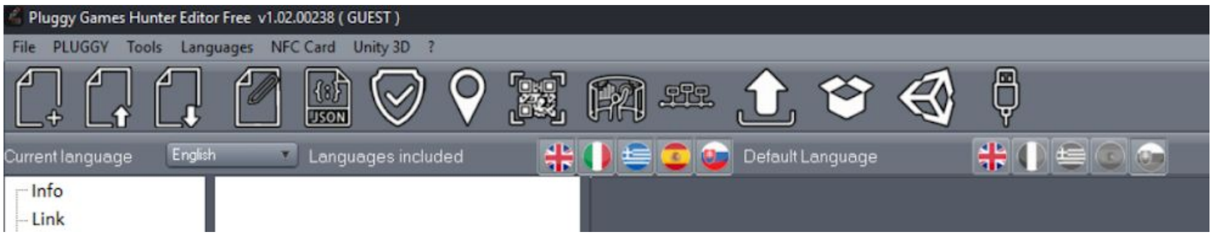
PLUGGY Login

Email:

Password: ************

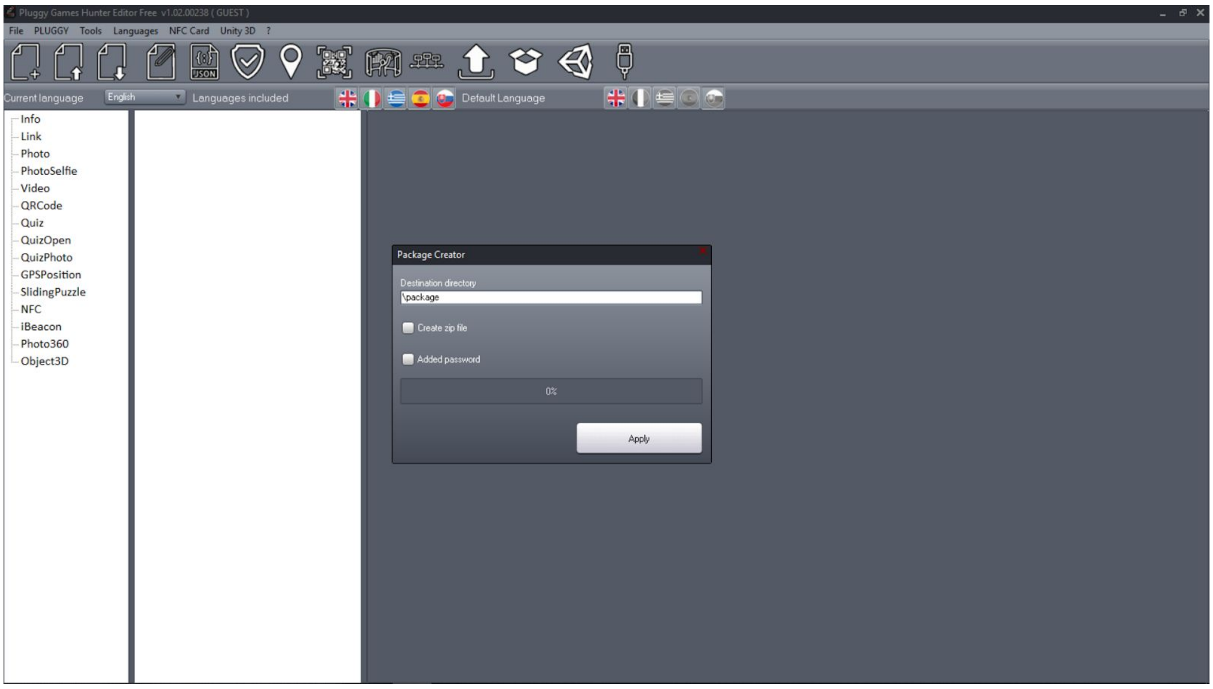
[✓] Remember me

Languages (default and included)


Pluggy Games Hunter Editor Free v1.02.00238 (GUEST)
File PLUGGY Tools Languages NFC Card Unity 3D ?
Current language
English
Languages included
Default Language
Info
Link

English	Italian	Greek	Spanish	Slovak

Create a package



Destination directory: \package

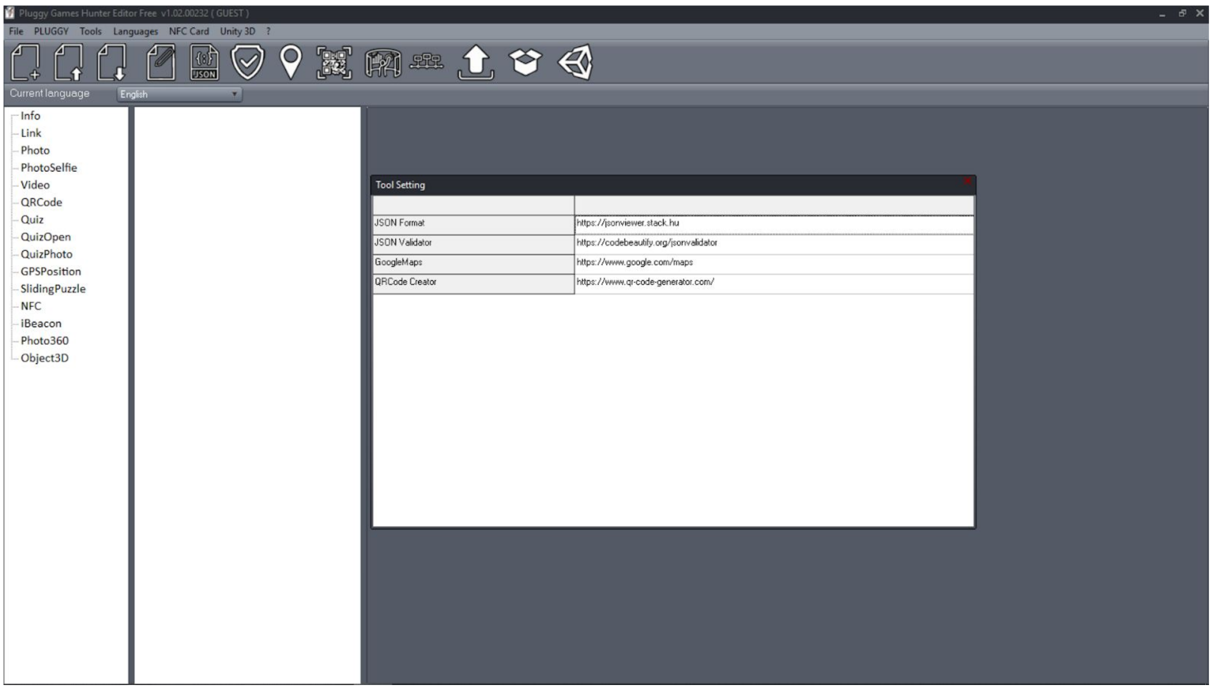
Create zip file

Added password

0%

Upload a package to PLUGGY

External Tool Setting



JSON Format	https://jsonviewer.stack.hu
JSON Validator	https://codebeautify.org/jsonvalidator
GoogleMaps	https://www.google.com/maps
QRCode Creator	https://www.qr-code-generator.com/