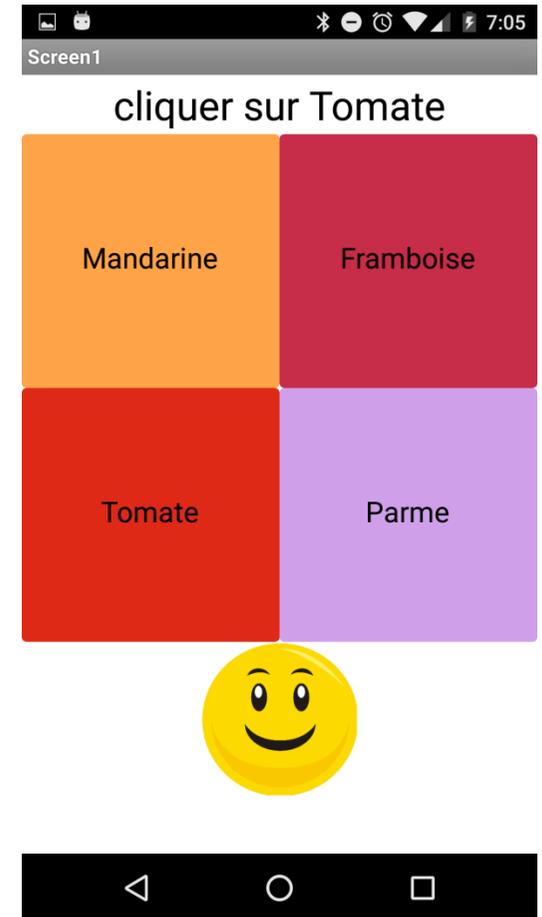
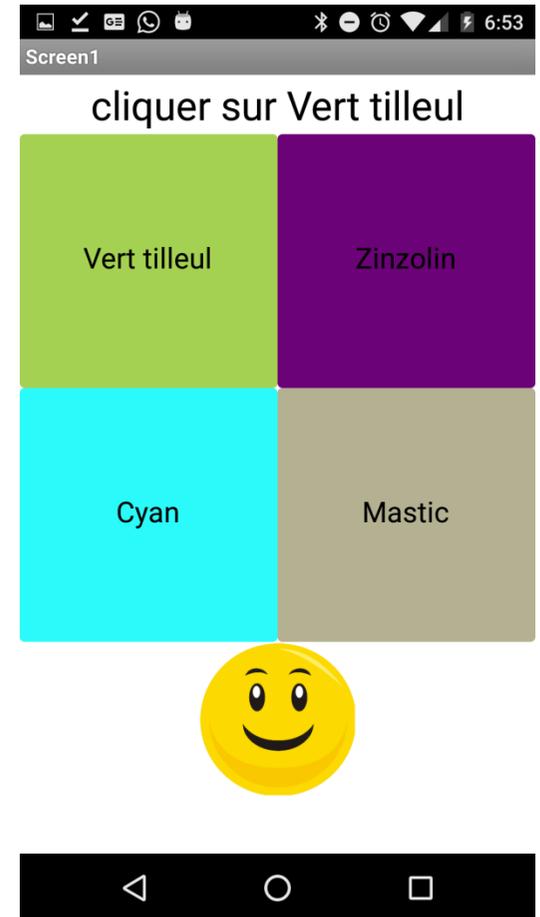


# LES COULEURS ET LEURS NOMS



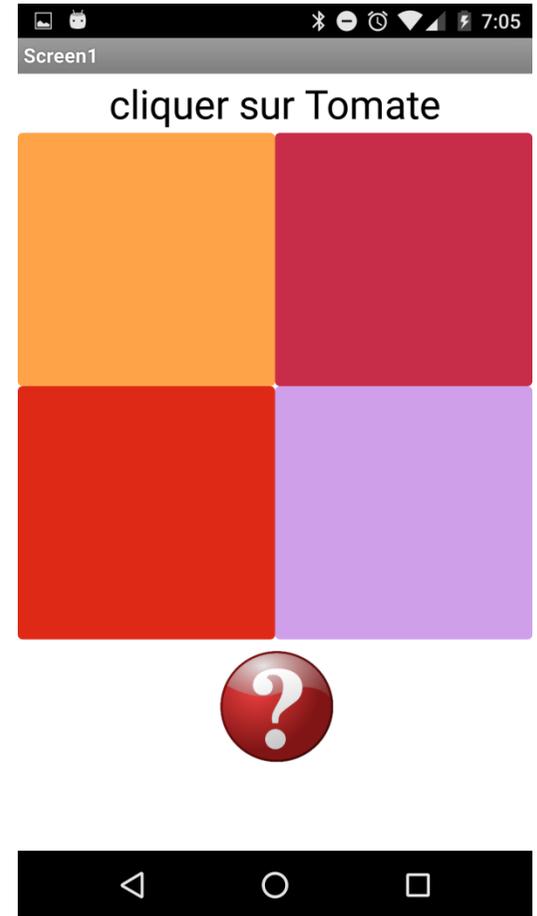
# LES COULEURS ET LEURS NOMS

- Objectifs :
  - Proposer une application pour reconnaître les couleurs
- Cible :
  - Parents : application pour jouer avec leurs jeunes enfants
- Principe :
  - V1 : présenter un nom de couleur, en afficher plusieurs et trouver la bonne
  - V2 : reconstruire une couleur à partir de ses composantes
- Données en entrée :
  - liste de couleurs définies par Wikipedia  
[https://fr.wikipedia.org/wiki/Liste\\_de\\_noms\\_de\\_couleur](https://fr.wikipedia.org/wiki/Liste_de_noms_de_couleur)



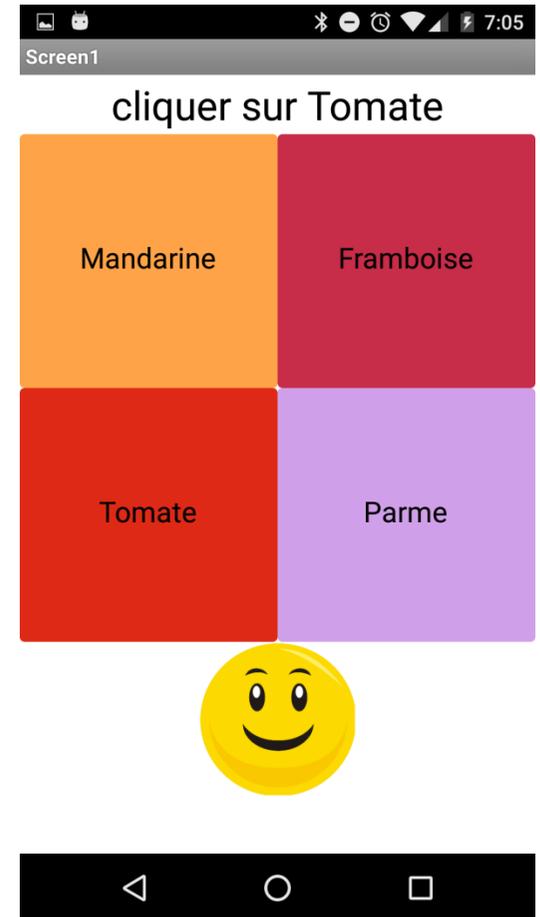
# PRINCIPE DU JEU

- 4 carrés de 4 couleurs prises au hasard  
(dans une liste wikipedia de 205 couleurs)  
en haut, le nom de la couleur à trouver ...



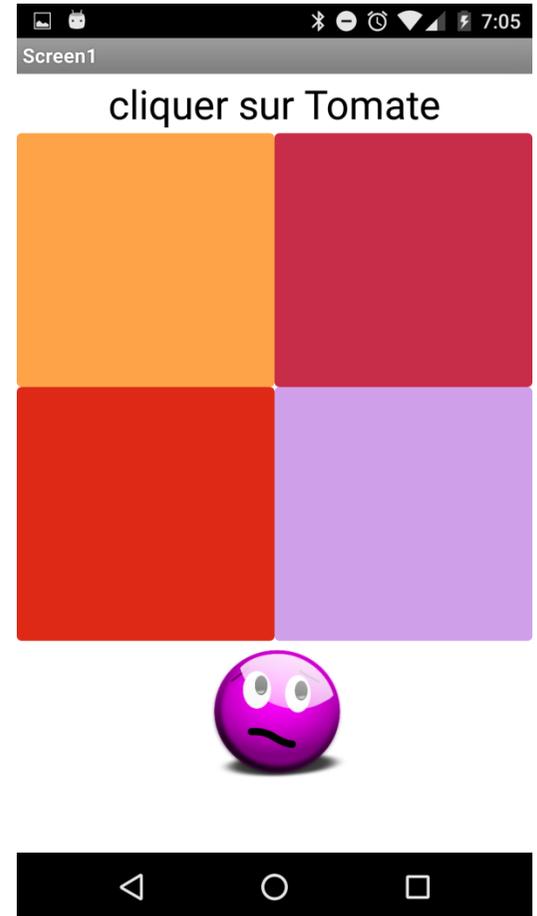
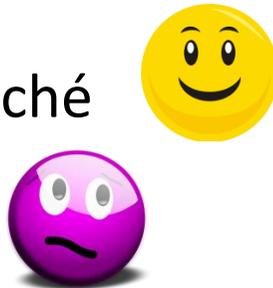
# PRINCIPE DU JEU

- 4 carrés de 4 couleurs prises au hasard  
(dans une liste wikipedia de 205 couleurs)  
en haut, le nom de la couleur à trouver ...
- L'utilisateur clique sur un carré
  - si c'est la bonne couleur :  
le nom des couleurs est affiché

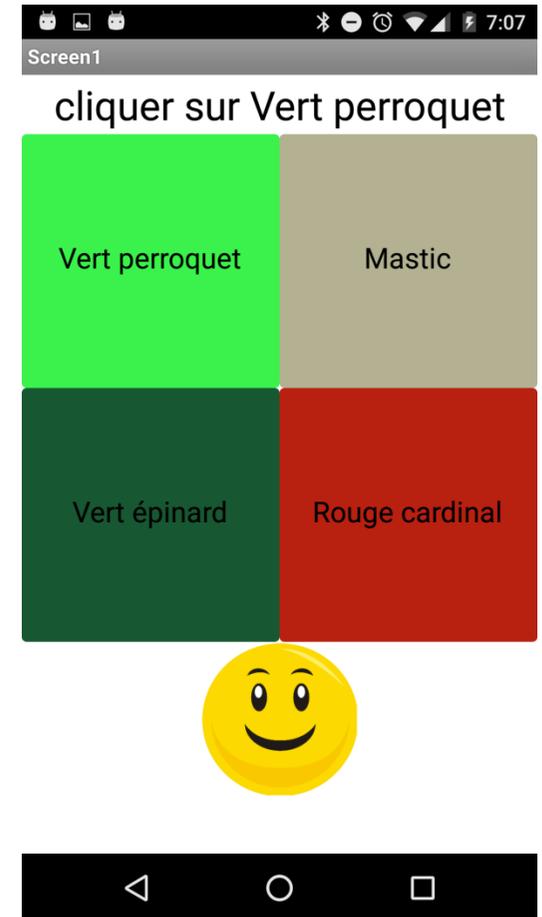


# PRINCIPE DU JEU

- 4 carrés de 4 couleurs prises au hasard (dans une liste wikipedia de 205 couleurs) en haut, le nom de la couleur à trouver ...
- l'utilisateur clique sur un carré
  - si c'est la bonne couleur : le nom des couleurs est affiché
  - sinon : faire un autre choix
- Clic sur l'icône pour un nouveau choix



# Développement de l'application



# LE PROGRAMME QUE L'ON VA RÉALISER : 115 À 130 BLOCS

```
initialise global listedeToutesLesCouleurs à créer une liste vide

quand Screen1 .Initialise
faire
mettre global listedeToutesLesCouleurs à appeler Web1 .Décoder Json Texte
appeler tirerAuSortEtAfficher4Couleurs

quand BoutonAfficherQuestion .Clic
faire
appeler tirerAuSortEtAfficher4Couleurs

à tirerAuSortEtAfficher4Couleurs
faire
mettre global listedes4IndexDeCouleurs à créer une liste vide
mettre global listedes4NomsDeCouleurs à créer une liste vide
mettre global listedes4RVBDeCouleurs à créer une liste vide
pour chaque choix de 1 à 4 par 1
faire
initialise local indexCouleurChoisiAuHasard à entier aléatoire entre 1 et taille de la liste liste obtenir global listedeToutesLesCouleurs
dans
initialise local (nomCouleur) à choisir liste élément liste obtenir global listedeToutesLesCouleurs
index obtenir indexCouleurChoisiAuHasard
initialise local (RVBCouleur) à choisir liste élément liste obtenir global listedeToutesLesCouleurs
index obtenir indexCouleurChoisiAuHasard
dans
ajouter éléments à la liste liste obtenir global listedes4IndexDeCouleurs
item obtenir indexCouleurChoisiAuHasard
ajouter éléments à la liste liste obtenir global listedes4NomsDeCouleurs
item choisir liste élément liste obtenir nomCouleur
index 1
ajouter éléments à la liste liste obtenir global listedes4RVBDeCouleurs
item choisir liste élément liste obtenir RVBCouleur
index 2
mettre global indexBonneReponseDansLes4 à entier aléatoire entre 1 et taille de la liste liste obtenir global listedes4IndexDeCouleurs
appeler afficher4Couleurs
```

## 3 procédures

- tirer au sort
- afficher
- vérifier la réponse

```
initialise global listedes4IndexDeCouleurs à créer une liste vide
initialise global listedes4NomsDeCouleurs à créer une liste vide
initialise global listedes4RVBDeCouleurs à créer une liste vide
initialise global indexBonneReponseDansLes4 à 0

quand Bouton1 .Clic
faire
appeler verifierReponse num 1

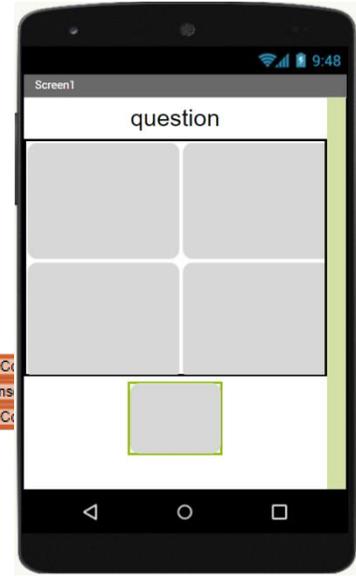
quand Bouton2 .Clic
faire
appeler verifierReponse num 2

quand Bouton3 .Clic
faire
appeler verifierReponse num 3

quand Bouton4 .Clic
faire
appeler verifierReponse num 4

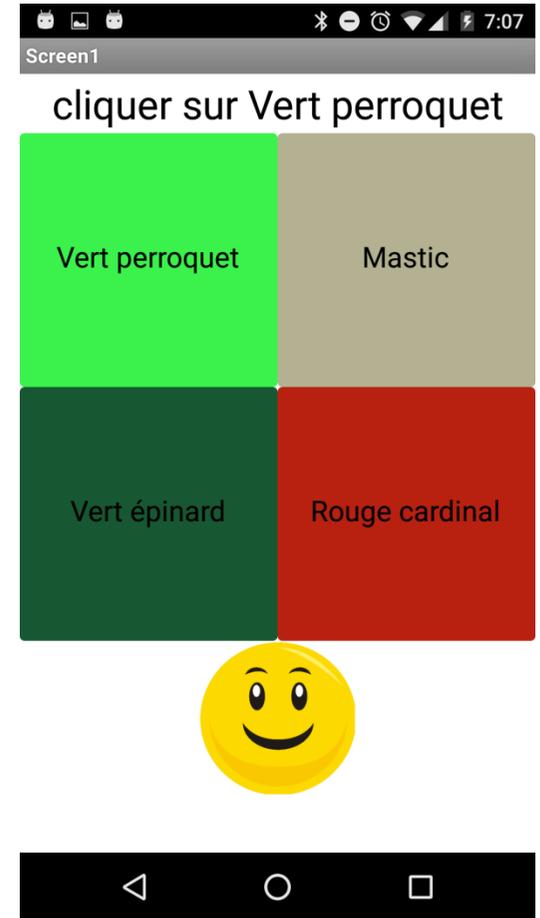
à verifierReponse num
faire
si comparer textes choisir liste élément liste obtenir global listedes4NomsDeCouleurs
index obtenir global indexBonneRepons
= choisir liste élément liste obtenir global listedes4NomsDeCouleurs
index obtenir num
alors mettre BoutonAfficherQuestion .Image à "smileyGagne.png"
sinon mettre BoutonAfficherQuestion .Image à "smileyPerdu.png"

à afficher4Couleurs
faire
mettre BoutonAfficherQuestion .Image à "question.png"
mettre LabelTexteQuestion .Texte à joint cliquer sur
choisir liste élément liste obtenir global listedes4NomsDeCouleurs
index obtenir global indexBonneReponseDansLes4
mettre LabelResultat .Texte à ""
mettre Bouton1 .Couleur de fond à créer couleur choisir liste élément liste obtenir global listedes4RVBDeCouleurs
index 1
mettre Bouton2 .Couleur de fond à créer couleur choisir liste élément liste obtenir global listedes4RVBDeCouleurs
index 2
mettre Bouton3 .Couleur de fond à créer couleur choisir liste élément liste obtenir global listedes4RVBDeCouleurs
index 3
mettre Bouton4 .Couleur de fond à créer couleur choisir liste élément liste obtenir global listedes4RVBDeCouleurs
index 4
```



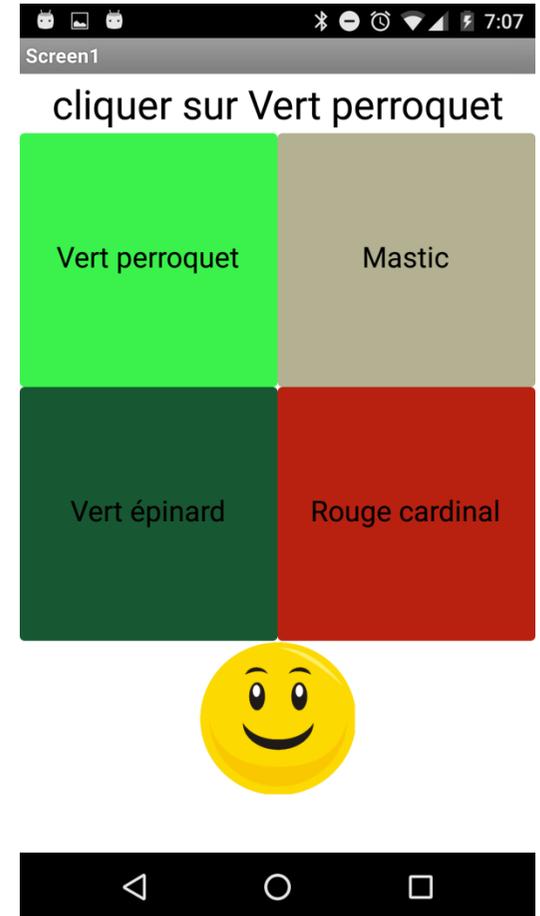
# NOTIONS TECHNIQUES UTILISÉES

- Notions de base que vous devez déjà maîtriser:
  - les variables
  - les boucles, les conditions
  - les procédures



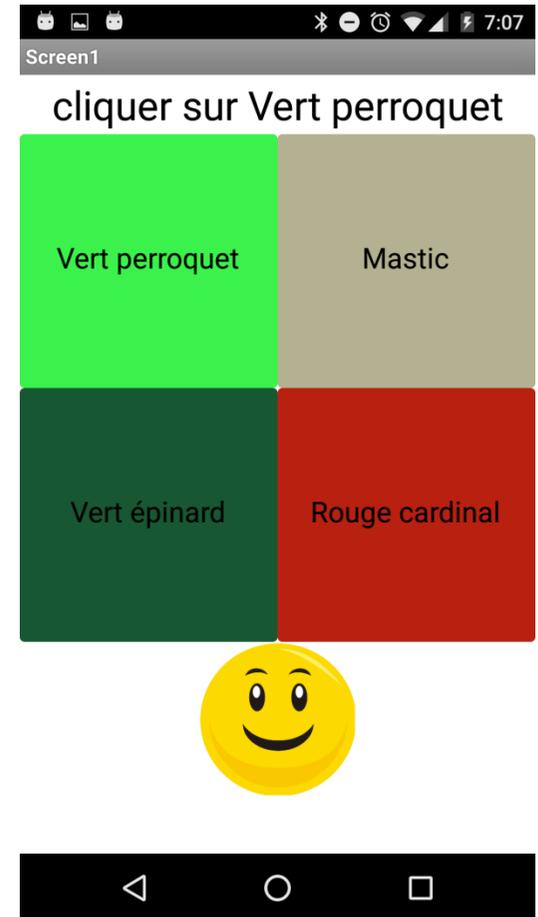
# NOTIONS TECHNIQUES UTILISÉES

- Notions de base que vous devez déjà maîtriser:
  - les variables
  - les boucles, les conditions
  - les procédures
- Notions délicates que vous devez avoir vues
  - les listes, les index et valeurs, les blocs de liste



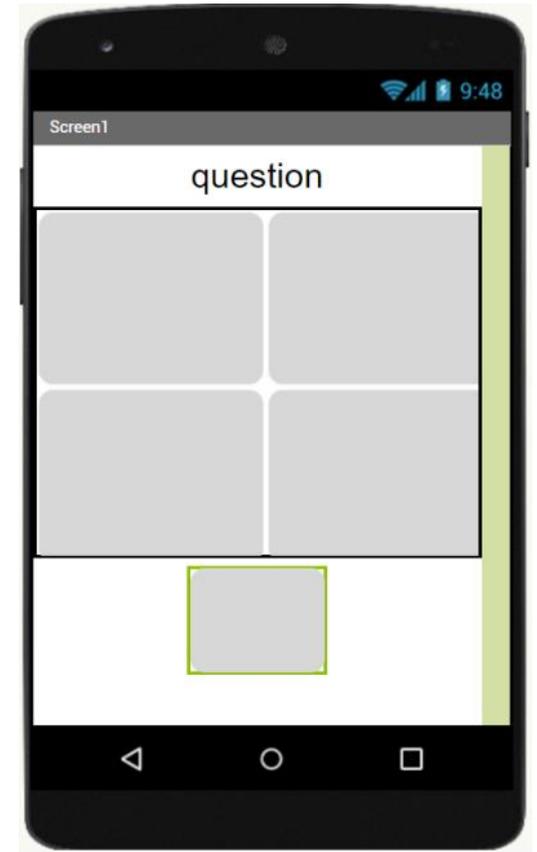
# NOTIONS TECHNIQUES UTILISÉES

- Notions de base que vous devez déjà maîtriser:
  - les variables
  - les boucles, les conditions
  - les procédures
- Notions délicates que vous devez avoir vues
  - les listes, les index et valeurs, les blocs de liste
- Notions plus difficiles
  - les listes de listes,
  - JSON pour décrire et organiser les données



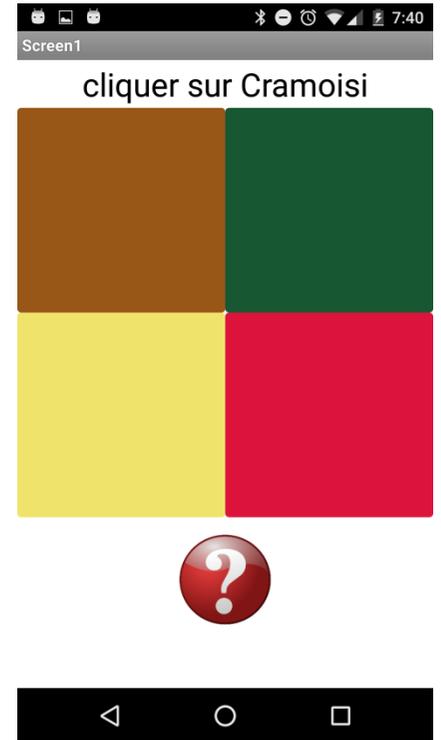
# DESIGN DE L'APPLICATION

- Design
  - un label pour afficher la couleur à trouver
  - un arrangement tableau 2 x 2, avec 4 boutons
  - un bouton pour afficher la réponse et proposer un nouveau choix de couleurs
  - Media :



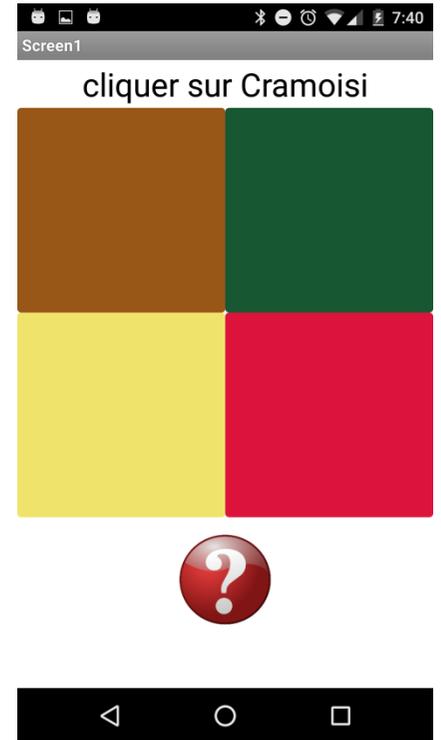
# ANALYSE ET CONCEPTION

- Evènements et actions associées :
  - au démarrage, lecture de la liste des couleurs pour la ranger dans la variable `listeDeToutesLesCouleurs`, puis appel de la procédure `tirerAuSortEtAfficher4couleurs`
  - quand clic sur bouton : appel à `tirerAuSortEtAfficher4couleurs`
  - quand clic sur un des 4 boutons de couleur, appel de la procédure `verificationReponse (n° bouton)`



# ANALYSE ET CONCEPTION

- Evènements et actions associées :
  - au démarrage, lecture de la liste des couleurs pour la ranger dans la variable `listeDeToutesLesCouleurs`, puis appel de la procédure `tirerAuSortEtAfficher4couleurs`
  - quand clic sur bouton : appel à `tirerAuSortEtAfficher4couleurs`
  - quand clic sur un des 4 boutons de couleur, appel de la procédure `verificationReponse (n° bouton)`
- Variables associées à chaque question
  - `listeDes4IndexDeCouleurs` liste des 4 index dans la liste des couleurs
  - `listeDes4NomsDeCouleurs` liste des 4 noms de couleurs tirées au sort
  - `listeDes4RVBDeCouleurs` liste des 4 composantes (chacune avec 3 valeurs)
  - `indexBonneReponseDansLes4` index de la bonne réponse (dans les 4 affichées)



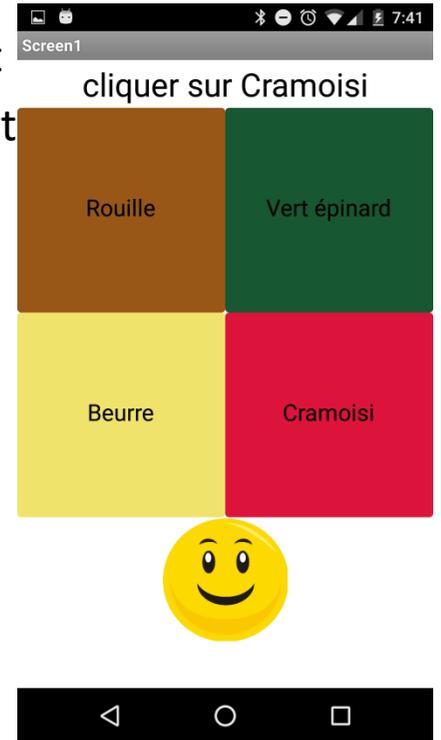
# ANALYSE ET CONCEPTION / PROCÉDURES ET VARIABLES

- Procédure `tirerAuSortEtAfficher4couleurs`

- reset des variables (index, nom, RVB) et réinitialisation par tirage au sort
- choix de l'index gagnant `indexBonneReponseDansLes4` par tirage au sort
- affichage des données de la question et des couleurs de boutons

- Procédure `afficher4couleurs`

- Affiche la couleur à trouver,
- Colore les boutons
- affiche 



# ANALYSE ET CONCEPTION / PROCÉDURES ET VARIABLES

- Procédure **tirerAuSortEtAfficher4couleurs**

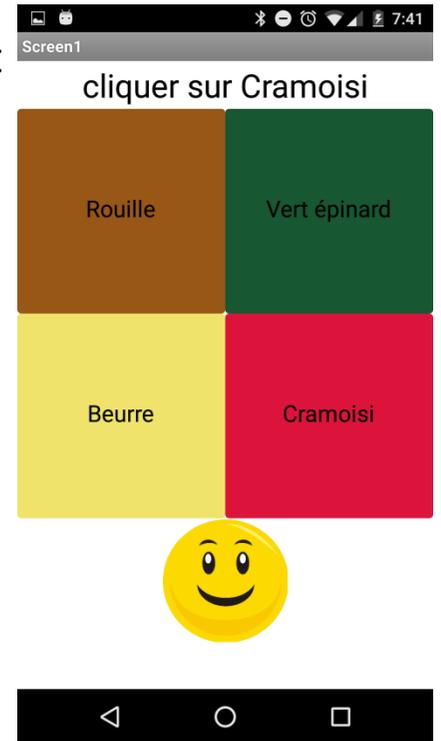
- reset des variables (index, nom, RVB) et réinitialisation par tirage au sort
- tirage au sort de l'index gagnant : **indexBonneReponseDansLes4**
- affichage des données de la question et des couleurs de boutons

- Procédure **afficher4couleurs**

- Affiche la couleur à trouver,
- Colore les boutons
- affiche 

- Procédure **verifierReponse (n°)**

- comparaison des couleurs de l'index gagnant et du bouton cliqué
- affichage de l'icone et des noms de couleurs selon la réponse



# Définition des couleurs

## Renseignement de la liste

### Données JSON

## DÉFINITION D'UNE COULEUR ?

Une couleur peut être définie comme une liste avec un nom et une liste de 3 composantes RVB :



## DÉFINITION D'UNE COULEUR ?

Une couleur peut être définie comme une liste avec un nom et une liste de 3 composantes RVB :



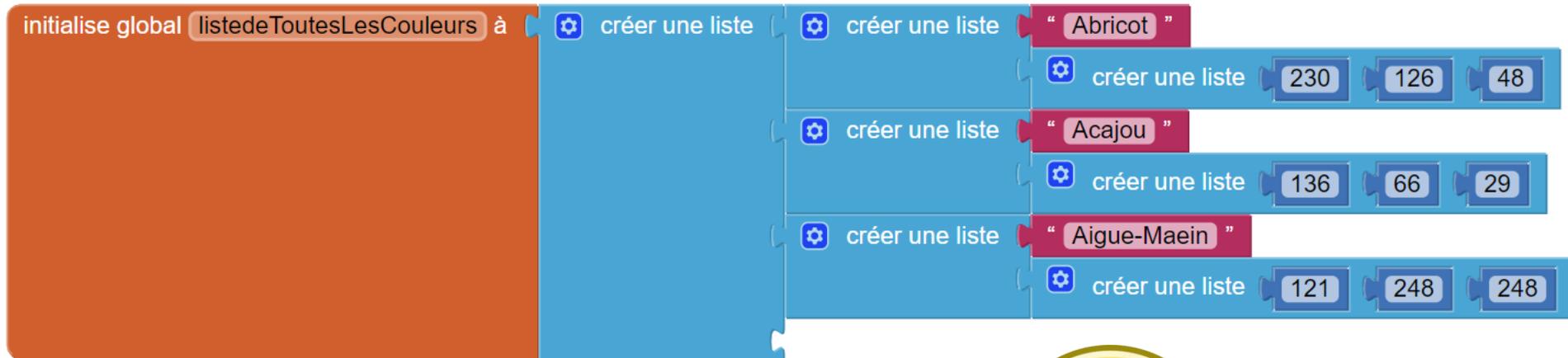
# LISTE DE TOUTES LES COULEURS ?

Puis on passe à la définition d'une liste de couleurs ...



## LISTE DE TOUTES LES COULEURS ?

Puis on passe à la définition d'une liste de couleurs ...  
mais avec avec Plus de 200 couleurs ... ça va être long



## DESCRIPTION DE LA LISTE AVEC LE FORMAT JSON

Dans le format JSON une liste est encadrée par des crochets et les éléments sont séparés par des virgules

```
[255, 0, 0]
```

## DESCRIPTION DE LA LISTE AVEC LE FORMAT JSON

Dans le format JSON une liste est encadrée par des crochets et les éléments sont séparés par des virgules

```
["rouge" , [255,0,0]]
```

## DESCRIPTION DE LA LISTE AVEC LE FORMAT JSON

Dans le format JSON une liste est encadrée par des crochets et les éléments sont séparés par des virgules

```
[  
  ["noir" , [0,0,0]],  
  ["rouge" , [255,0,0]],  
  ["vert" , [0,255,0]],  
  ["bleu" , [0,0,255]],  
  ["cyan" , [0,255,255]],  
  ["Magenta" , [255,0,255]],  
  ["jaune" , [255,255,0]],  
  ["Blanc" , [255,255,255]]  
]
```

# DÉCODAGE DU FORMAT JSON EN LISTES APP INVENTOR

Dans le format JSON une liste est encadrée par des crochets et les éléments sont séparés par des virgules

```
[  
  ["noir" , [0,0,0]],  
  ["rouge" , [255,0,0]],  
  ["vert" , [0,255,0]],  
  ["bleu" , [0,0,255]],  
  ["cyan" , [0,255,255]],  
  ["Magenta" , [255,0,255]],  
  ["jaune" , [255,255,0]],  
  ["Blanc" , [255,255,255]]  
]
```

Et on convertit ces données en listes App Inventor



The screenshot shows an App Inventor block with the following configuration:

- mettre** `global listedeToutesLesCouleurs` à **appeler** `Web1` `.Décoder Json Texte`
- Texte Json** : `[["Abricot" , [230,126,48]], ["Acajou" , [136,6... ]]`

# DÉCODAGE DU FORMAT JSON EN LISTES APP INVENTOR



Le bloc précédent (pour 8 couleurs) est équivalent à :

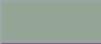


# APPLICATION À 205 COULEURS PROPOSÉES PAR WIKIPEDIA :

- La liste des noms et composantes RVB est extraite dans un tableur

1	Nom de couleur	Échantillon n	Code RVB		
2			R	V	B
3	<a href="#">Abricot</a>		230	126	48
4	<a href="#">Acajou</a>		136	66	29
5	<a href="#">Aigue-marine</a>		121	248	248
6	<a href="#">Alezan (chevaux)</a>		167	103	38
7	<a href="#">Amande</a>		130	196	108

...

205	<a href="#">Vert-de-gris</a>		149	165	149
206	<a href="#">Violet d'évêque</a>		114	62	100
207	<a href="#">Viride</a>		64	130	109
208	<a href="#">Zinzolin</a>		108	2	119

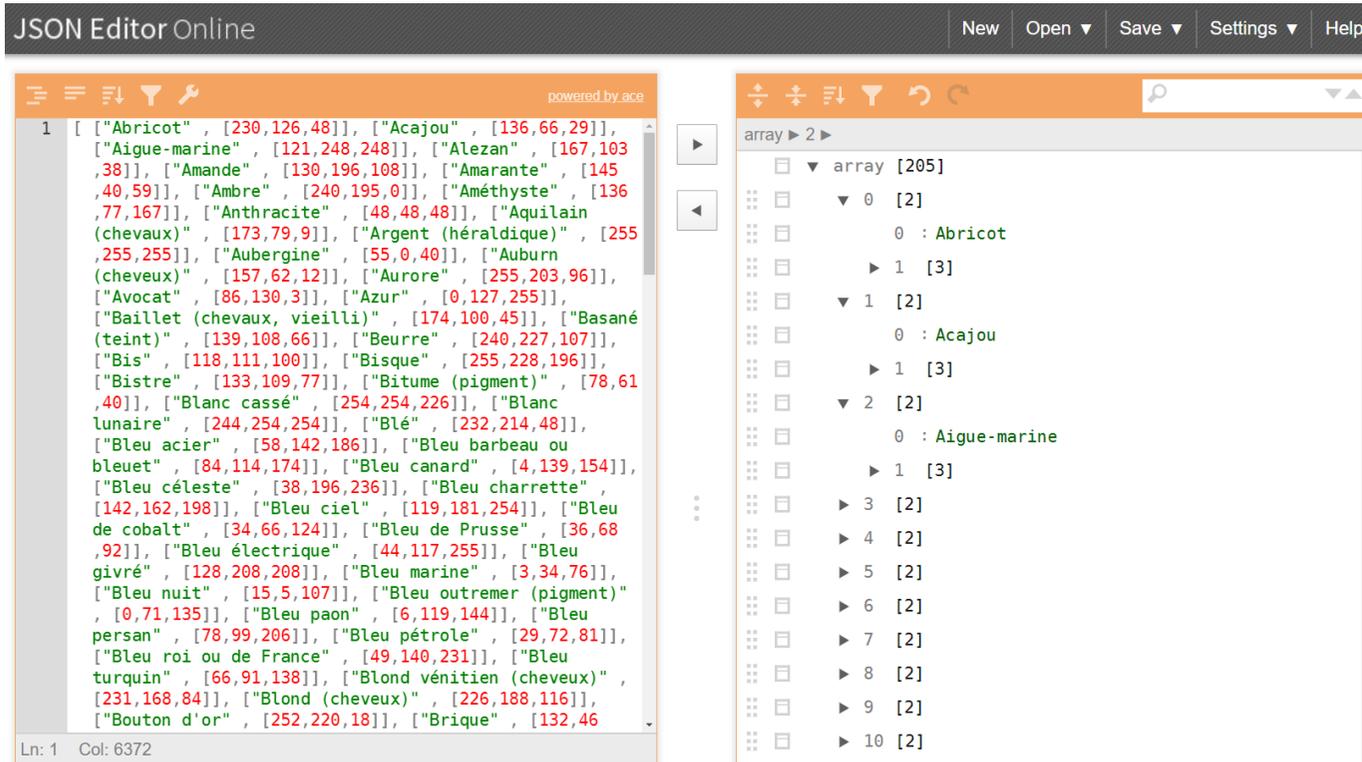
# APPLICATION À 205 COULEURS PROPOSÉES PAR WIKIPEDIA :

- Convertie au format JSON avec crochets et séparateurs

1	Nom de couleur	Échantillon	Code RVB								
			R	V	B						
3	<a href="#">Abricot</a>		230	126	48	[	["	<a href="#">" ,</a>	]	,	[["Abricot" , [230,126,48]],
4	<a href="#">Acajou</a>		136	66	29	[	["	<a href="#">" ,</a>	]	,	["Acajou" , [136,66,29]],
5	<a href="#">Aigue-marine</a>		121	248	248	[	["	<a href="#">" ,</a>	]	,	["Aigue-marine" , [121,248,248]],
6	<a href="#">Alezan (chevaux)</a>		167	103	38	[	["	<a href="#">" ,</a>	]	,	["Alezan (chevaux)" , [167,103,38]],
7	<a href="#">Amande</a>		130	196	108	[	["	<a href="#">" ,</a>	]	,	["Amande" , [130,196,108]],
...											
205	<a href="#">Vert-de-gris</a>		149	165	149	[	["	<a href="#">" ,</a>	]	,	["Vert-de-gris" , [149,165,149]],
206	<a href="#">Violet d'évêque</a>		114	62	100	[	["	<a href="#">" ,</a>	]	,	["Violet d'évêque" , [114,62,100]],
207	<a href="#">Viride</a>		64	130	109	[	["	<a href="#">" ,</a>	]	,	["Viride" , [64,130,109]],
208	<a href="#">Zinzolin</a>		108	2	119	[	["	<a href="#">" ,</a>	]	]	["Zinzolin" , [108,2,119]]

# APPLICATION À 205 COULEURS PROPOSÉES PAR WIKIPEDIA :

- Puis vérification avec un outil ( internet) comme JSON Editor online



The screenshot shows the JSON Editor Online interface. The left pane displays a JSON array of 205 color names and their associated RGB values. The right pane shows a tree view of the array, with the first few elements expanded to show their structure.

```
1 [ ["Abricot" , [230,126,48]], ["Acajou" , [136,66,29]],  
  ["Aigue-marine" , [121,248,248]], ["Alezan" , [167,103,38]],  
  ["Amande" , [130,196,108]], ["Amarante" , [145,40,59]],  
  ["Ambre" , [240,195,0]], ["Améthyste" , [136,77,167]],  
  ["Anthracite" , [48,48,48]], ["Aquilain (chevaux)" , [173,79,9]],  
  ["Argent (héraldique)" , [255,255,255]], ["Aubergine" , [55,0,40]],  
  ["Auburn (cheveux)" , [157,62,12]], ["Aurore" , [255,203,96]],  
  ["Avocat" , [86,130,3]], ["Azur" , [0,127,255]],  
  ["Baillet (chevaux, vieilli)" , [174,100,45]], ["Basané (teint)" , [139,108,66]],  
  ["Beurre" , [240,227,107]], ["Bis" , [118,111,100]], ["Bisque" , [255,228,196]],  
  ["Bistre" , [133,109,77]], ["Bitume (pigment)" , [78,61,40]],  
  ["Blanc cassé" , [254,254,226]], ["Blanc lunaire" , [244,254,254]],  
  ["Blé" , [232,214,48]], ["Bleu acier" , [58,142,186]],  
  ["Bleu barbeau ou bleu" , [84,114,174]], ["Bleu canard" , [4,139,154]],  
  ["Bleu céleste" , [38,196,236]], ["Bleu charrette" , [142,162,198]],  
  ["Bleu ciel" , [119,181,254]], ["Bleu de cobalt" , [34,66,124]],  
  ["Bleu de Prusse" , [36,68,92]], ["Bleu électrique" , [44,117,255]],  
  ["Bleu givré" , [128,208,208]], ["Bleu marine" , [3,34,76]],  
  ["Bleu nuit" , [15,5,107]], ["Bleu outremer (pigment)" , [0,71,135]],  
  ["Bleu paon" , [6,119,144]], ["Bleu persan" , [78,99,206]],  
  ["Bleu pétrole" , [29,72,81]], ["Bleu roi ou de France" , [49,140,231]],  
  ["Bleu turquin" , [66,91,138]], ["Blond vénitien (cheveux)" , [231,168,84]],  
  ["Blond (cheveux)" , [226,188,116]], ["Bouton d'or" , [252,220,18]],  
  ["Brique" , [132,46
```

<https://jsoneditoronline.org/>

# Programme :

Au démarrage lecture des données JSON  
et appel à la procédure de  
tirage au sort et d'affichage

## MISE EN ŒUVRE DANS LE PROGRAMME :

“ [ ["Abricot" , [230,126,48]], ["Acajou" , [136,6... ] ] ”

[  
 [ "Abricot", [ 230, 126, 48 ] ],  
 [ "Acajou", [ 136, 66, 29 ] ],  
 [ "Aigue-marine", [ 121, 248, 248 ] ],  
 ...  
 [ "Viride", [ 64, 130, 109 ] ],  
 [ "Zinzolin", [ 108, 2, 119 ] ]  
]



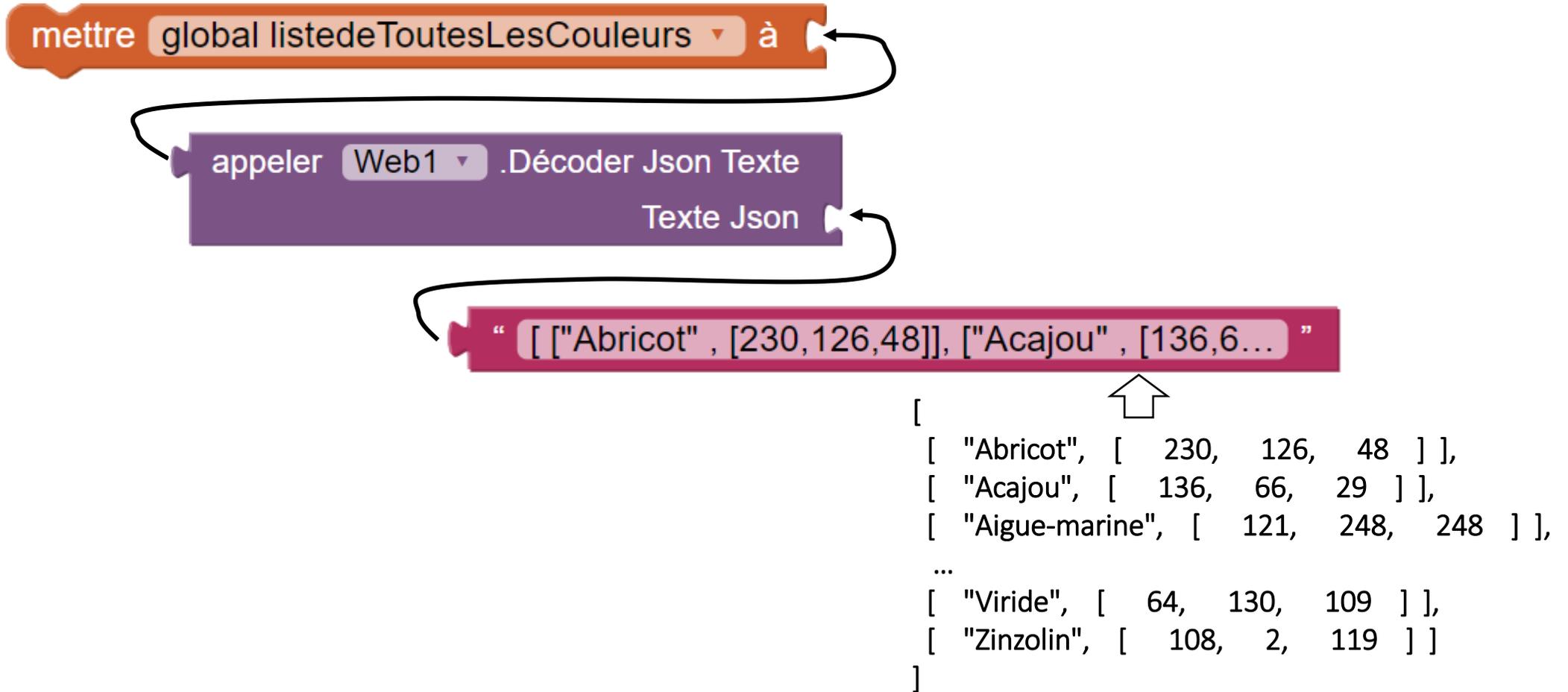
## MISE EN ŒUVRE DANS LE PROGRAMME :

appeler Web1 .Décoder Json Texte  
Texte Json

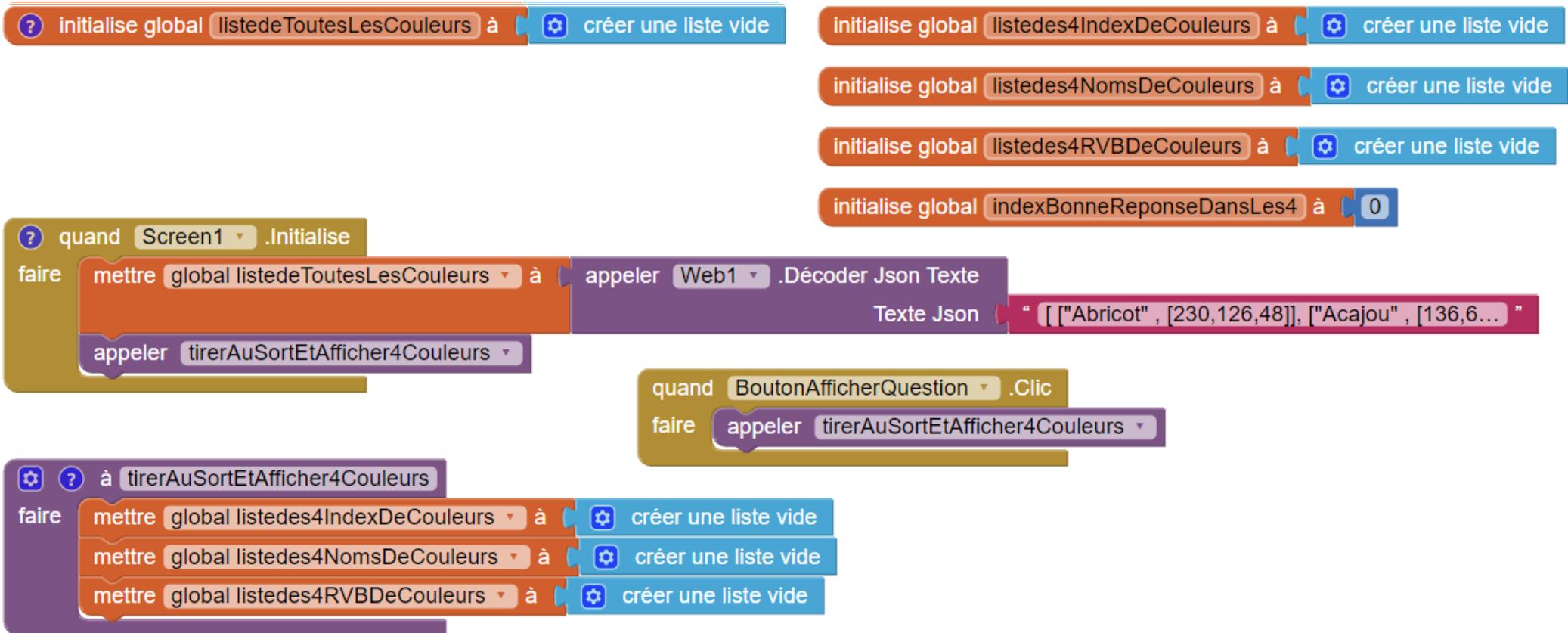
“ [ ["Abricot" , [230,126,48]], ["Acajou" , [136,6...

```
[  
  [ "Abricot", [ 230, 126, 48 ] ],  
  [ "Acajou", [ 136, 66, 29 ] ],  
  [ "Aigue-marine", [ 121, 248, 248 ] ],  
  ...  
  [ "Viride", [ 64, 130, 109 ] ],  
  [ "Zinzolin", [ 108, 2, 119 ] ]  
]
```

## MISE EN ŒUVRE DANS LE PROGRAMME :

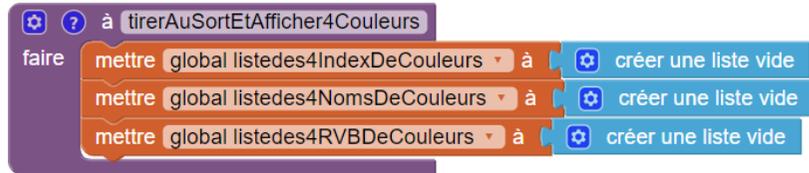


# DÉMARRAGE DE L'APPLICATION

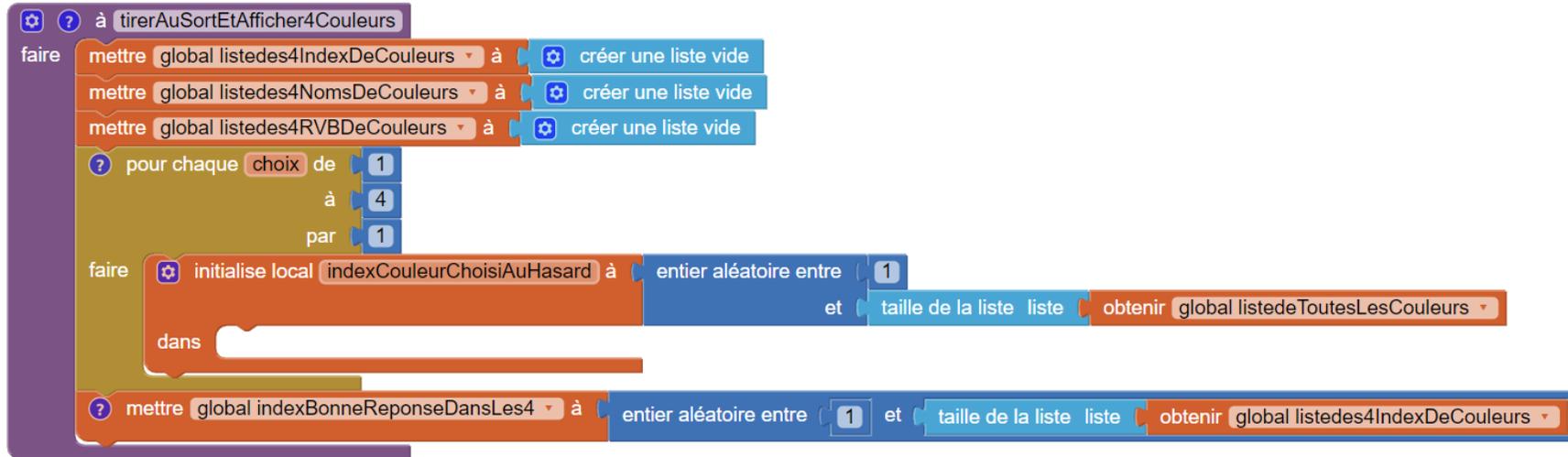


# Procédure(s) de tirage au sort et d'affichage

# PROCÉDURE DE TIRAGE AU SORT



# PROCÉDURE DE TIRAGE AU SORT



```
à tirerAuSortEtAfficher4Couleurs
faire
  mettre global listedes4IndexDeCouleurs à créer une liste vide
  mettre global listedes4NomsDeCouleurs à créer une liste vide
  mettre global listedes4RVBDeCouleurs à créer une liste vide
  pour chaque choix de 1
    à 4
    par 1
    faire
      initialise local indexCouleurChoisiAuHasard à entier aléatoire entre 1
      et taille de la liste liste obtenir global listedeToutesLesCouleurs
      dans
    mettre global indexBonneReponseDansLes4 à entier aléatoire entre 1
    et taille de la liste liste obtenir global listedes4IndexDeCouleurs
```

The image shows a Scratch script for a procedure named "tirerAuSortEtAfficher4Couleurs". The script starts with a "faire" loop that contains three "mettre" blocks to initialize global lists: "listedes4IndexDeCouleurs", "listedes4NomsDeCouleurs", and "listedes4RVBDeCouleurs", each set to "créer une liste vide". This is followed by a "pour chaque" loop with "choix" from 1 to 4, with a "par" block set to 1. Inside this loop, there is a "faire" loop that starts with "initialise local" for "indexCouleurChoisiAuHasard" set to "entier aléatoire entre 1" and "taille de la liste liste" obtained from "global listedeToutesLesCouleurs". The "dans" block of this inner loop is currently empty. Finally, there is a "mettre" block for "global indexBonneReponseDansLes4" set to "entier aléatoire entre 1" and "taille de la liste liste" obtained from "global listedes4IndexDeCouleurs".

# PROCÉDURE DE TIRAGE AU SORT

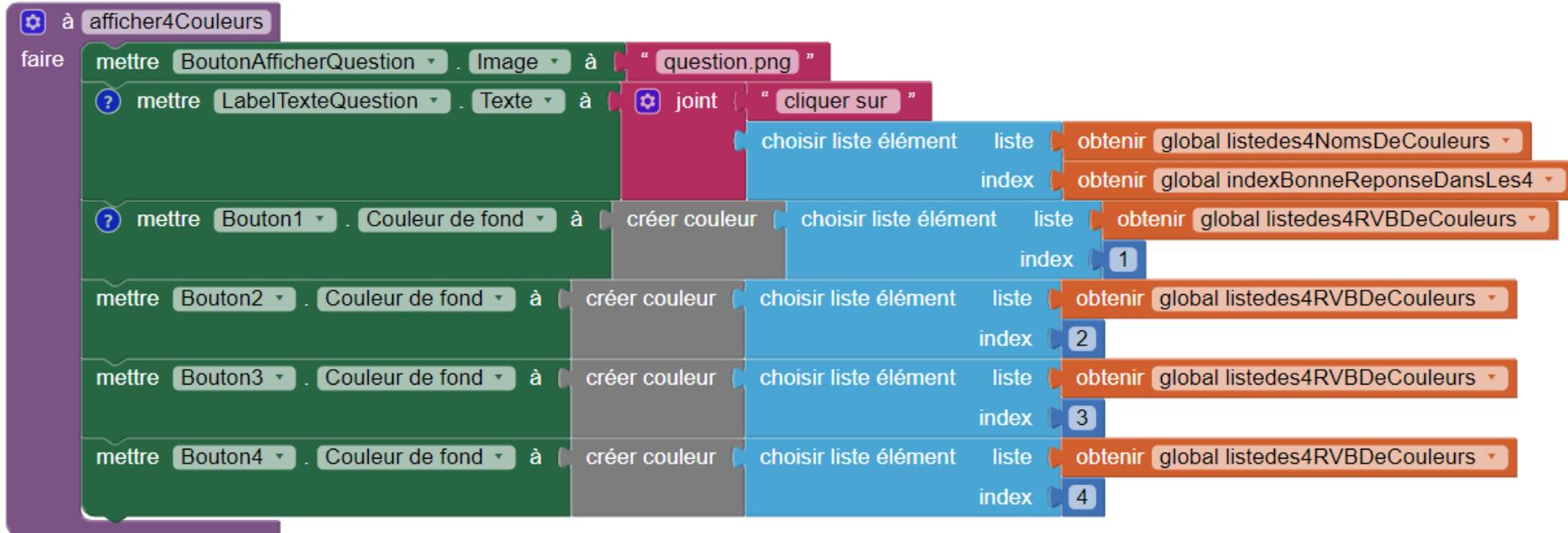
```
à tirerAuSortEtAfficher4Couleurs
faire
  mettre global listedes4IndexDeCouleurs à créer une liste vide
  mettre global listedes4NomsDeCouleurs à créer une liste vide
  mettre global listedes4RVBDeCouleurs à créer une liste vide
  pour chaque choix de 1
    à 4
    par 1
      faire
        initialise local indexCouleurChoisiAuHasard à entier aléatoire entre 1
          et taille de la liste liste obtenir global listedeToutesLesCouleurs
        dans
          initialise local couleur à choisir liste élément liste obtenir global listedeToutesLesCouleurs
            index obtenir indexCouleurChoisiAuHasard
        dans
        dans
      faire
        mettre global indexBonneReponseDansLes4 à entier aléatoire entre 1
          et taille de la liste liste obtenir global listedes4IndexDeCouleurs
    appel afficher4Couleurs
```

# PROCÉDURE DE TIRAGE AU SORT

The image shows a Scratch script for a procedure named "tirerAuSortEtAfficher4Couleurs". The script is as follows:

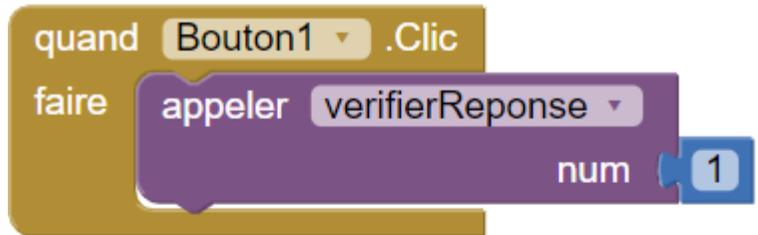
```
faire
  mettre global listedes4IndexDeCouleurs à créer une liste vide
  mettre global listedes4NomsDeCouleurs à créer une liste vide
  mettre global listedes4RVBDeCouleurs à créer une liste vide
  pour chaque choix de 1
    à 4
    par 1
      faire
        initialise local indexCouleurChoisiAuHasard à entier aléatoire entre 1
          et taille de la liste liste obtenir global listede ToutesLesCouleurs
        dans
          initialise local couleur à choisir liste élément liste obtenir global listede ToutesLesCouleurs
            index obtenir indexCouleurChoisiAuHasard
          dans
            ajouter éléments à la liste liste obtenir global listedes4IndexDeCouleurs
              item obtenir indexCouleurChoisiAuHasard
            ajouter éléments à la liste liste obtenir global listedes4NomsDeCouleurs
              item choisir liste élément liste obtenir couleur
                index 1
            ajouter éléments à la liste liste obtenir global listedes4RVBDeCouleurs
              item choisir liste élément liste obtenir couleur
                index 2
          fin
        fin
      fin
    fin
  fin
  mettre global indexBonneReponseDansLes4 à entier aléatoire entre 1
    et taille de la liste liste obtenir global listedes4IndexDeCouleurs
  appeler afficher4Couleurs
```

# PROCÉDURE D’AFFICHAGE APPELÉE À LA FIN DU TIRAGE AU SORT



# Choix et vérification de la réponse

# FONCTION DE VÉRIFICATION APPELÉE AVEC LE N° DU BOUTON



# FONCTION DE VÉRIFICATION APPELÉE AVEC LE N° DU BOUTON

```
quand Bouton1 .Clic  
faire appeler verifierReponse  
num 1
```

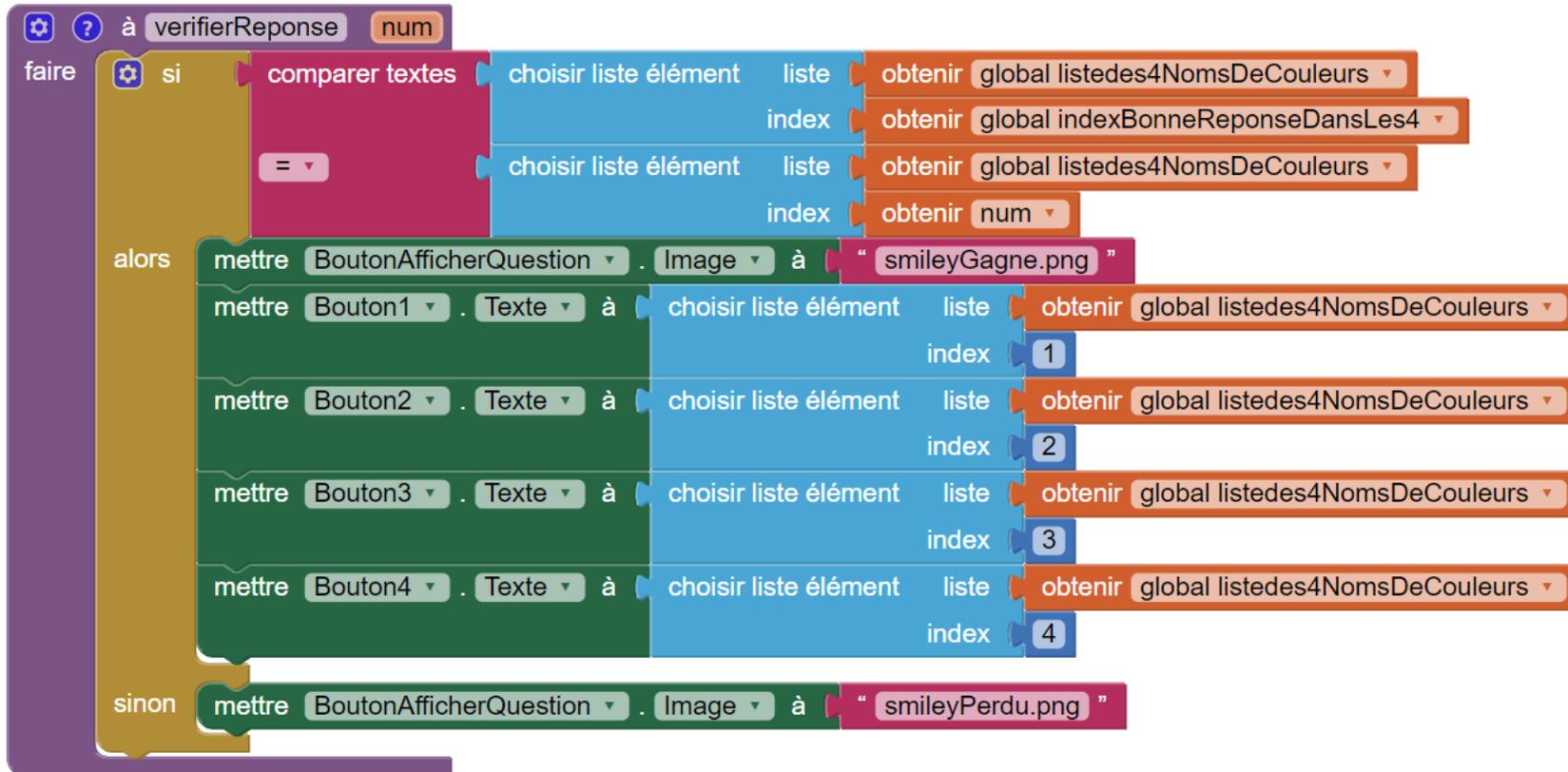
```
quand Bouton2 .Clic  
faire appeler verifierReponse  
num 2
```

```
quand Bouton3 .Clic  
faire appeler verifierReponse  
num 3
```

```
quand Bouton4 .Clic  
faire appeler verifierReponse  
num 4
```

```
à verifierReponse num  
faire
```

# FONCTION / PROCÉDURE DE VÉRIFICATION

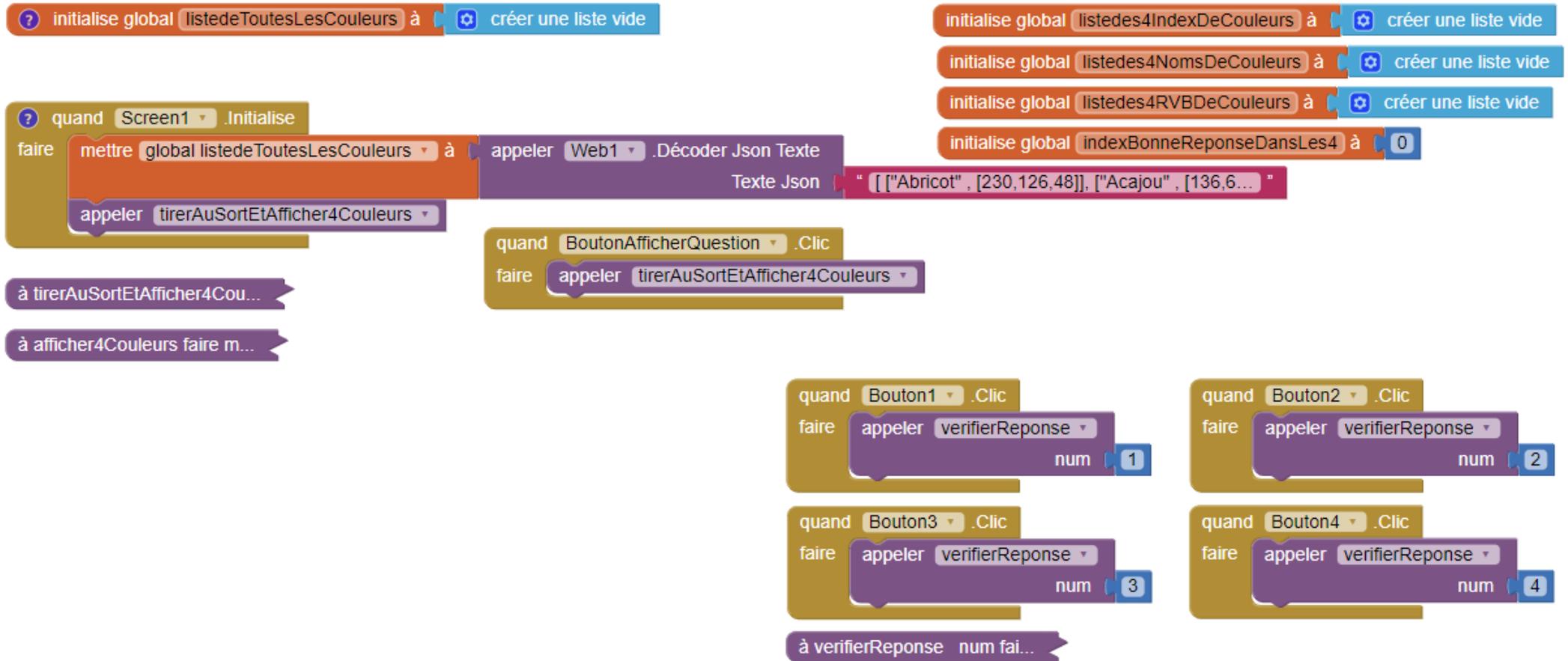


# PROGRAMME RÉALISÉ

The image displays a collection of Scratch code blocks for a game titled 'Couleurs du monde'. The code is organized into several functional sections:

- Initialization:** Global variables for color lists and a response index are set up. A screen initialization block triggers the loading of JSON data and the random selection of colors and names.
- Question Generation:** A loop that generates a question by randomly selecting a color and a name, then concatenating them into a text label.
- Buttons:** Four buttons are created, each with a unique background color and a text label corresponding to one of the selected colors.
- Click Events:** Each button has a click event that calls a 'verify response' function with its index (1-4).
- Verification:** A function that compares the user's selected color with the correct one. It updates the 'BoutonAfficherQuestion' image to either 'smileyGagne.png' (win) or 'smileyPerdu.png' (lose).

# PROGRAMME RÉALISÉ



Fin

# NOMS DE COULEURS ET COMPOSANTES RVB (WIKIPEDIA)

Liste [ [modifier](#) | [modifier le code](#) ]

Nom de couleur	Échantillon	Code hexadécimal			Code RVB		
		#	R	V	B	R	V
Abricot		# E6	7E	30	230	126	48
Acajou		# 88	42	1D	136	66	29
Aigue-marine		# 79	F8	F8	121	248	248
Alezan (chevaux)		# A7	67	26	167	103	38
Amande		# 82	C4	6C	130	196	108
Amarante		# 91	28	3B	145	40	59
Ambre		# F0	C3	00	240	195	0
Améthyste		# 88	4D	A7	136	77	167
Anthracite		# 30	30	30	48	48	48
Aquilain (chevaux)		# AD	4F	09	173	79	9
Argent (héraldique)		# FF	FF	FF	255	255	255
Aubergine		# 37	00	28	55	0	40

# LES COULEURS : NOMS ET COMPOSANTES RVB

Une couleur peut être définie par son nom et ses composantes RVB : Rouge, Vert, Bleu

parfois complétées par un 4<sup>e</sup> canal de transparence (canal alpha)

Par exemple, pour les couleurs saturées :

Nom de couleur	Échantillon	R	V	B
noir		0	0	0
rouge		255	0	0
vert		0	255	0
bleu		0	0	255
cyan		0	255	255
Magenta		255	0	255
jaune		255	255	0
Blanc		255	255	255

# LES COULEURS : NOMS ET COMPOSANTES RVB

Une couleur peut être définie par son nom et ses composantes RVB : Rouge, Vert, Bleu

parfois complétées par un 4° canal de transparence (canal alpha)

Ou la liste de 205 noms [proposés par Wikipedia](#):

Nom de couleur	Échantillon	Code RVB		
		R	V	B
Abricot		230	126	48
Acajou		136	66	29
Aigue-marine		121	248	248
Alezan (chevaux)		167	103	38
Amande		130	196	108
Amarante		145	40	59
Ambre		240	195	0
Améthyste		136	77	167
Anthracite		48	48	48

## REPRÉSENTATION DES COULEURS DANS APP INVENTOR ?

On va plutôt utiliser une description textuelle : JSON

```
[  
  ["noir" , [0,0,0]],  
  ["rouge" , [255,0,0]],  
  ["vert" , [0,255,0]],  
  ["bleu" , [0,0,255]],  
  ["cyan" , [0,255,255]],  
  ["Magenta" , [255,0,255]],  
  ["jaune" , [255,255,0]],  
  ["Blanc" , [255,255,255]]  
]
```

# LES COULEURS : NOMS ET COMPOSANTES RVB

Une couleur peut être définie par son nom  
et par ses composantes RVB : Rouge, Vert, Bleu  
parfois complétées par un 4<sup>e</sup> canal de transparence (canal alpha)

Par exemple, pour les couleurs saturées :

Nom de couleur	Échantillon	R	V	B
noir		0	0	0
rouge		255	0	0
vert		0	255	0
bleu		0	0	255
cyan		0	255	255
Magenta		255	0	255
jaune		255	255	0
Blanc		255	255	255

## LES COULEURS : NOM ET COMPOSANTES RVB

Une couleur peut être définie par son nom  
et par ses composantes RVB : Rouge, Vert, Bleu  
parfois complétées par un 4° canal de transparence (canal alpha)

Par exemple, pour les couleurs saturées :

Nom de couleur	Échantillon	R	V	B	description JSON
					[
noir		0	0	0	["noir" , [0,0,0]],
rouge		255	0	0	["rouge" , [255,0,0]],
vert		0	255	0	["vert" , [0,255,0]],
bleu		0	0	255	["bleu" , [0,0,255]],
cyan		0	255	255	["cyan" , [0,255,255]],
Magenta		255	0	255	["Magenta" , [255,0,255]],
jaune		255	255	0	["jaune" , [255,255,0]],
Blanc		255	255	255	["Blanc" , [255,255,255]]
					]