2016/2017

Jean Morales

Lycée François d'Estaing

Informatique et Sciences du Numérique

Dossier-projet

Type du projet	Editeur photo
Nom du projet	PHOTOS&FILTERS
Membres de l'équipe	Jean Morales, Adrien Josquin

Choix de l'équipe

Mon souhait était d'appartenir à un groupe restreint, avec peu de personnes. Pour moi, il y a plus de travail personnel à fournir dans un groupe restreint. Je pense qu'un groupe de deux personnes est plus facile à organiser, notamment dans la répartition des tâches. En revanche, il faut bien connaître son binôme. Je connais Adrien depuis la classe de 6ème. C'est la personne avec qui je partage mon enthousiasme pour l'ISN. Mon choix de binôme s'est donc logiquement dirigé vers lui. De plus, nous habitons proches l'un de l'autre, ce qui facilite la mise en commun de nos travaux hors cours d'ISN.

Choix du projet

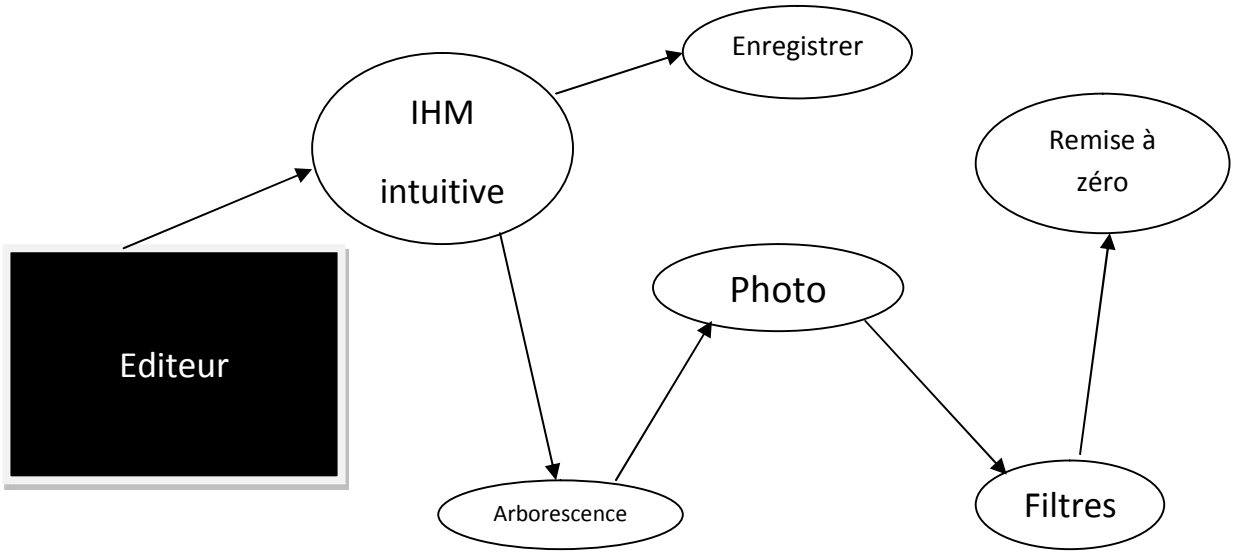
Compte tenu de nos affinités, Adrien et moi sommes vite tombés d'accord. Nous pensions en début d'année à la réalisation d'un jeu de type « Space Invaders ». Dans ce jeu, l'utilisateur doit faire survivre son personnage à plusieurs vagues d'ennemis. Puis notre avis a changé. En effet, le cours sur les images et les pixels nous a vraiment plu. Nous avons donc décidé, à partir de ce moment là, que nous ferions un éditeur photo.

Le choix du langage

Le choix du langage de programmation nous a paru simple. Python est un langage à notre portée. Les modules complémentaires sont fournis avec EduPython. Lorsque l'on installe EduPython, on peut directement programmer. De plus, nous savions que nous l'utiliserions toute l'année. Ainsi, dès le début d'année, nous savions que nous travaillerions avec Python pour ce projet. Pour ma part, je n'ai pas longtemps hésité avec le Visual Basic, que j'ai commencé à apprendre durant les vacances d'été précédentes. Je trouve Python plus intuitif. Nous avons donc choisi EduPython pour coder notre éditeur photo.

Les objectifs du projet

Nous voulions réaliser un éditeur photo. Cet éditeur peut appliquer des filtres à une image. Une image doit être sélectionnée par l'utilisateur. Il va rechercher cette image grâce à un menu, où s'affiche l'arborescence de l'ordinateur. L'interface homme-machine, ou IHM, est intuitive. L'utilisateur peut revenir en arrière s'il le souhaite. Tous les formats d'image peuvent être supportés. L'image choisie est visible à l'écran, quelle que soit sa taille.



La répartition des tâches

Je trouve Tkinter plaisant à utiliser. Tkinter est un module de Python, utilisé pour les interfaces graphiques. Je le trouve simple d'utilisation et intuitif. Pour ma part, je me suis attribué le travail lié à l'interface homme-machine. Par exemple, j'ai en charge les boutons apparaissant à l'écran, ainsi que l'image qui va s'actualiser après la sélection d'un filtre.

Adrien sait comment modifier une image en lui appliquant des fonctions. Il s'est alors approprié la partie « Filtres », et s'est chargé d'obtenir la documentation nécessaire à nos besoins sur le programme.

Nous travaillons chacun de manière autonome, et effectuons un bilan sur notre avancée chaque semaine.

Les difficultés rencontrées

Nos objectifs nous ont semblé modestes. Nous avons donc dû rajouter des options à notre programme, pour qu'il soit plus élaboré. C'est durant cette partie que nous avons rencontré des difficultés. En effet, nous voulions faire apparaître le flux d'une webcam à l'écran, puis prendre une photo.

J'ai ensuite pensé à ajouter des barres de défilement. Ces outils permettraient de modifier en temps réel les valeurs du rouge, vert et bleu de l'image affichée. C'est la partie la plus complexe du programme. Nous avons réussi à les mettre en œuvre, mais nous ne sommes pas satisfaits de leur action. Ainsi, nous avons bien trois barres de défilement, mais elles ne modifient pas les valeurs du rouge, vert et bleu comme nous le voudrions.

Le résultat obtenu

Nous avons fabriqué un éditeur photo. On peut choisir une image de n'importe quel format, et l'ouvrir, l'afficher à l'écran, quelque soit sa taille, à l'aide d'un bouton. Avec d'autres boutons, on peut quitter le programme, appliquer des filtres à l'image affichée, et annuler tous les filtres appliqués. On aura le poids de l'image ouverte qui sera affiché, et qui s'actualise à chaque nouvelle image ouverte. L'image affichée sera délimitée par un cadre, une mire noire. On peut bien sûr appliquer plusieurs filtres en même temps à une même image. Une image est affichée sur chaque bouton. La fenêtre du programme possède une icône, et affiche le nom de notre éditeur. L'utilisateur peut enregistrer son image sous le format PNG. Le programme est rapide dans l'exécution des tâches qui lui sont demandées.

Les améliorations possibles

Des améliorations peuvent être apportées à notre programme. La première chose à améliorer est la modification du rouge, vert et bleu de l'image avec les barres de défilement, ou scrollbars. On pourrait afficher des informations supplémentaires sur l'image, comme son format. Ensuite, il faudrait que l'utilisateur puisse choisir sous quel format il veut enregistrer son image. Enfin, on pourrait brancher une caméra à l'ordinateur et faire apparaître son flux vidéo en temps réel, en pouvant prendre des photos. On peut aussi penser à afficher une image de présentation du programme.

Mon rôle au final

Je me suis occupé de l'aspect visuel du programme. J'ai créé les boutons, la mire de l'image, les scrollbars. Pour ces dernières, je me suis chargé de leurs fonctions associées, avec l'aide d'Adrien. J'ai affecté à chaque bouton une image. J'ai trouvé et affiché le logo de notre programme en haut à gauche de la fenêtre.

J'ai fait en sorte que l'utilisateur puisse aller chercher une image dans l'ordinateur, quelle que soit son format ou sa taille. J'ai créé les boutons « Remise à zéro » et « Sauvegarder ». Ils permettent respectivement d'annuler tous les filtres utilisés sur l'image, et de sauver l'image créée par l'utilisateur. J'ai enfin élaboré quelques filtres. J'ai conçu les filtres « Sépia » et « Noir et Blanc ».

Bilan de ce que j'ai appris

Cette réalisation de projet m'a appris de nombreuses choses. J'ai dû apprendre à imaginer un produit fini que je serai capable de concevoir. Pour cela il faut prendre conscience de la motivation de chacun, de notre capacité à travailler de manière autonome et régulière. Je sais désormais qu'il n'y a pas de solution unique à un problème informatique. La résolution d'un problème dépend de notre capacité à nous remettre en question, de notre motivation, et de notre implication dans la matière. J'ai appris à organiser mon travail dans le temps, et à collaborer avec un binôme de manière régulière.

Annexe

Sites consultés :

- OpenClassroom

- Developpez.com

- CodeLab

- Stack Overflow

Vient ensuite ma partie du programme :

```
8  ############################ FILTRES ########################################
9
10 def sepia () :
11     if compteur == 1 :
12         global image , photo
13         pixel = image.load ()
14         for x in range (image.size [0]) :
15             for y in range (image.size [1]) :
16
17                 red = pixel [x,y] [0]
18                 gris = int ( (pixel [x,y] [0] + pixel [x,y] [1] + pixel [x,y] [2]) / 3)
19                 red = int (1.25 * red)
20
21                 pixel [x,y] = (red , gris , gris)
22
23         image.save ('En cours.png')
24         photo = ImageTk.PhotoImage (image)
25         canvas1.create_image (0 , 0 , anchor = NW , image = photo)
26 """ Moyenne la valeur des composantes et affecte cette moyenne au vert et au bleu, puis intensifie le rouge """
27
28 def flou () :
```

```
""" Fais une moyenne des composantes RGB, puis applique cette moyenne à chaque pixel """

def nb () :
    if compteur == 1 :
        global image , photo
        pixel = image.load ()
        for x in range (image.size [0]) :
            for y in range (image.size [1]) :

                red = pixel [x,y] [0]
                green = pixel [x,y] [1]
                blue = pixel [x,y] [2]

                gris = int ( (pixel [x,y] [0] + pixel [x,y] [1] + pixel [x,y] [2]) / 3)
                if gris <= 175 :
                    red = 0
                    green = 0
                    blue = 0
                else :
                    red = 255
                    green = 255
                    blue = 255

                pixel [x,y] = (red , green , blue)

        image.save ('En cours.png')
        photo = ImageTk.PhotoImage (image)
        canvas1.create_image (0 , 0 , anchor = NW , image = photo)
""" Définition d'un seuil. En dessous de ce seuil, le pixel est blanc. Sinon il est noir """

def brg () :
```

```
""" Inverse la couleur de chaque pixel """

############################## MODIFICATION COMPOSANTES RGB ######################

def b (a) :
    if compteur == 1 :
        global image , photo , r , g , b

        b = image.load()

        z = value3.get()

        for v in range (image.size [0]) :
            for w in range (image.size [1]) :

                blue1 = b [v,w] [0]
                blue = int()
                if blue1 + (blue1 * z) / 100 > 255 :
                    blue = 255
                if blue1 + (blue1 * z) / 100 < 0 :
                    blue = 0

                blue = int(blue1 + (blue1 * z) / 100)

                b [v,w] = (b [v,w] [0] , b [v,w] [1] , blue)


        image.save ('En cours.png')
        photo = ImageTk.PhotoImage (image)
        canvas1.create_image (0 , 0 , anchor = NW , image = photo)
""" Modifie la composante bleu de l'image, est associée à la scrollbar 'Bleu' """
```

```
157 """ Modifie la composante bleu de l'image, est associée à la scrollbar 'Bleu' """
158
159 def g (a) :
160     if compteur == 1 :
161         global image , photo , r , g , b
162         g = image.load()
163
164         y = value2.get()
165
166         for v in range (image.size [0]) :
167             for w in range (image.size [1]) :
168
169                 green1 = g [v,w] [1]
170                 green = int()
171                 if green1 + (green1 * y) / 100 > 255 :
172                     green = 255
173                 if green + (green * y) / 100 < 0 :
174                     green = 0
175
176                 green = int(green1 + (green1 * y) / 100)
177
178                 g [v,w] = (g [v,w] [0] , green , g [v,w] [2])
179
180
181         image.save ('En cours.png')
182         photo = ImageTk.PhotoImage (image)
183         canvas1.create_image (0 , 0 , anchor = NW , image = photo)
184 """ Modifie la composante vert de l'image, , est associée à la scrollbar 'Vert' """
185
186 def r (a) :
187     if compteur == 1 :
```

```
184 """ Modifie la composante vert de l'image, , est associée à la scrollbar 'Vert' """
185
186 def r (a) :
187     if compteur == 1 :
188         global image , photo , r , r1
189
190         r = image.load ()
191         r1 = image.load ()
192
193         x = value1.get()
194
195         for v in range (image.size [0]) :
196             for w in range (image.size [1]) :
197
198                 red1 = r1 [v,w] [0]
199                 red = r [v,w] [0]
200
201                 if red + (red * x) / 100 > 255 :
202                     red = 255
203
204                 if red + (red * x) / 100 < 0 :
205                     red = 0
206
207                 if x == 0 :
208                     r [v,w] = (r1 [v,w] [0] , r1 [v,w] [1] , r1 [v,w] [2])
209
210                 red = int (red + (red * x) / 100)
211
212                 r [v,w] = (red , r [v,w] [1] , r [v,w] [2])
213
214
215
216         image.save ('En cours.png')
217         photo = ImageTk.PhotoImage (image)
218         canvas1.create_image (0 , 0 , anchor = NW , image = photo)
219 """ Modifie la composante rouge de l'image """
220
```

```
220
221 ############################ PROCEDURES DE BASE ##############################
222
223 def quit () :
224     image = Image.open ('Quit.png')
225     image.save ('En cours.png')
226     fen.destroy ()
227 """ Renc le fichier 'En cours.png' vierge et ferme la fenêtre """
228
229 def save () :
230     global image , dimensions
231     image = image.resize (dimensions)
232     filename = asksaveasfile (mode = 'w' , title = 'Enregistrer sous' , filetypes = [ ('all files' , '*') , ('all files','*') ] , defaultextension = '.png')
233     image.save (filename.name)
234
235 def retour () :
236     if compteur == 1 :
237         global image , photo
238         image = Image.open ('Retour.png')
239         image.save ('En cours.png')
240         photo = ImageTk.PhotoImage (image)
241         canvas1.create_image (0 , 0 , anchor = NW , image = photo)
242 """ Affiche une copie de l'image de base """
243
244 def canvas () :
245     global size
246     canvas1.configure (width = 740 , height = size[1] - 10)
247 """ Réajuste le caneves pour éviter que la bordure du bas soit trop grosse """
248
249
250 def label () :
251     global data , resolution
252     resolution.destroy ()
253     str = 'POIDS:' , int ((len (data) / 1024) + 1) , "Ko"
254     resolution = Label (Panel , text = str)
255     resolution.grid (row = 1 , column = 2)
256 """ Affiche le poids de l'image pour chaque image affichée"""
```

```
275 def arborescence () :
276     global image , photo , dimensions , size , data , compteur , r , g , b
277     filepath = askopenfilename (title = 'Ouvrir une image' , filetypes = [ ('all files' , '*') , ('all files','*') ] , multiple = False)
278     image = Image.open (filepath)
279     dimensions = image.size
280     image = image.resize ( (750 , int ((750 / image.size [0]) *image.size [1] ) ) )
281     size = image.size
282     image.save ('En cours.png')
283     image.save ('Retour.png')
284     photo = ImageTk.PhotoImage (image)
285     canvas1.create_image (0 , 0 , anchor = NW , image = photo)
286     r,g,b = image.split()
287     r.save('r.jpg') #Sauvegarde des différents images composantes
288     g.save('g.jpg')
289     b.save('b.jpg')
290
291     copie ()
292
293     if photo.height () < 500 :
294         canvas1.pack (pady = 150)
295     elif photo.height () < 750 :
296         canvas1.pack (pady = 75)
297     else :
298         canvas1.pack (pady = 0)
299
300     pixel = image.load()
301
302     compteur = 1
303
304     FichierSource = open (filepath ,'rb')
305     data = [int (i) for i in FichierSource.read ()]
306
307     label ()
308
309     canvas ()
310
311     canvas1.configure (bg = 'black')
```

```
314
315 ######################### CREATION FENETRES ####################################
316
317 fen = Tk ()
318 fen.iconbitmap ('title.ico')
319 fen.title ('Photos&Filters')
320
321 Panel = Frame (fen)
322 Panel.pack (side = 'left')
323
324 ######################### PREPARATION IMAGE ####################################
325
326 compteur = 0
327
328 image = Image.open ('En cours.png')
329 image = image.resize ( (750 , int ( ( 750 / image.size [0]) * image.size [1] ) ) )
330 photo = ImageTk.PhotoImage (image)
331
332 canvas1 = Canvas (fen , width = photo.width () , height = photo.height () , bd = 7)
333 canvas1.pack ()
334
335 ################## RECUPERATION DES COMPOSANTES RGB ############################
336
337 value1 = IntVar ()
338 scale1 = Scale (Panel , variable = value1 , from_= -100 , to = 100 , resolution = 25 , command = r)
339 scale1.grid (row = 2 , column = 0 , pady = 20)
340
341 value2 = IntVar ()
342 scale2 = Scale (Panel , variable = value2 , from_= -100 , to = 100 , resolution = 25 , command = g)
343 scale2.grid (row = 2 , column = 1)
344
345 value3 = IntVar ()
346 scale3 = Scale (Panel , variable = value3 , from_= -100 , to = 100 , resolution = 25 , command = b)
347 scale3.grid (row = 2 , column = 2)
348
349 ############################## BOUTONS #########################################
350
```

```
347 scale3.grid (row = 2 , column = 2)
348
349 ############################## BOUTONS #########################################
350
351 im1 = ImageTk.PhotoImage(file = "f:/ISN/ZZ--PROJET FINAL/Modules/V6/OUVRIR.png")
352 B_img = Button (Panel , image = im1 , cursor = 'hand1' , command = arborescence , height = 100 , width = 100)
353 B_img.grid (padx = 50 , row = 0 , column = 0)
354
355 im2 = ImageTk.PhotoImage(file = 'f:/ISN/ZZ--PROJET FINAL/Modules/V6/REMISE A ZERO.png')
356 B_rtr = Button (Panel , text = 'REMISE A ZERO' , image = im2 , cursor = 'hand1' , command = retour , height = 100 , width = 100)
357 B_rtr.grid (pady = 10 , padx = 10 , row = 1 , column = 1)
358
359 im3 = ImageTk.PhotoImage(file = 'f:/ISN/ZZ--PROJET FINAL/Modules/V6/SAUVER.png')
360 B_sv = Button (Panel , text = 'SAUVER' , image = im3 , cursor = 'hand1' , command = save , height = 100 , width = 100)
361 B_sv.grid (padx = 10 , row = 0 , column = 1)
362
363 im4 = ImageTk.PhotoImage(file = 'f:/ISN/ZZ--PROJET FINAL/Modules/V6/QUITTER.png')
364 B_qtr = Button (Panel , text = 'QUITTER' , image = im4 , cursor = 'hand1' , command = quit , height = 100 , width = 100)
365 B_qtr.grid (pady = 20 , padx = 50 , row = 0 , column = 2)
366
367 im6 = ImageTk.PhotoImage(file = 'f:/ISN/ZZ--PROJET FINAL/Modules/V6/INFO.png')
368 resolution = Label (Panel , image = im6)
369 resolution.grid (row = 1 , column = 2)
370
371 rouge = Label (Panel , text = '     Rouge     ' , bg = 'red')
372 rouge.grid (row = 3 , column = 0)
373
374 vert = Label (Panel , text = '      Vert     ' , bg = 'green')
375 vert.grid (row = 3 , column = 1)
376
377 bleu = Label (Panel , text = '      Bleu     ' , bg = 'cyan')
378 bleu.grid (row = 3 , column = 2)
379
380 ######################### MENU DES FILTRES #####################################
```

```
bleu.grid (row = 3 , column = 2)

########################## MENU DES FILTRES ###################################

im5 = ImageTk.PhotoImage(file = 'f:/ISN/ZZ--PROJET FINAL/Modules/V6/FILTRES.png')
mb = Menubutton (Panel , text = 'FILTRES' , image = im5 , cursor = 'hand1' , height = 100 , width = 100 , relief = GROOVE)
mb.grid (pady = 20 , row = 1 , column = 0)

mb.menu = Menu (mb , tearoff = 0)
mb ['menu'] = mb.menu

mb.menu.add_command (label = 'FLOU' , command = flou)
mb.menu.add_command (label = 'INVERSION DES BANDES' , command = brg)
mb.menu.add_command (label = 'NEGATIF' , command = negatif)
mb.menu.add_command (label = 'MONOCHROME' , command = mono)
mb.menu.add_command (label = 'NIVEAUX DE GRIS' , command = ng)
mb.menu.add_command (label = 'NOIR ET BLANC' , command = nb)
mb.menu.add_command (label = 'SEPIA' , command = sepia)
mb.menu.add_command (label = 'SYMETRIE HORIZONTALE' , command = sh)
mb.menu.add_command (label = 'SYMETRIE VERTICALE' , command = sa)

########################## STABILITE DE LA FENETRE TKINTER ####################

fen.mainloop ()
```