dongguk UNIVERSITY

REPORT

과 목 명 : 스크립트프로그래밍01

제 출 일 : 06월 01일

학 번 : 2020213311

성 명 : 신중근

목차

1 소개

이 코드는 이미지 편집을 위한 프로그램입니다. 프로그램은 tkinter 라이브러리를 사용하여 GUI 환경에서 작동하며, 사용자는 이미지 파일을 선택하고 다양한 이미지 편집 작업을 수행할 수 있습니다. 편집 기능에는 이미지 필터링, 명암 조절, 반전 및 회전 등 다양한 기능이 구현되어 있습니다. 또한, 사용자가 작업한 이미지는 SQLite 데이터베이스에 저장되어 추후에도 접근할 수 있도록 합니다.

2 기능 설명

2.1 화면 표시

```
11  def displayImage(img, width, height):
12      global window, canvas, paper, photo, photo2, oriX, oriY
13
14      window.geometry(str(width) + "x" + str(height))
15      if canvas is not None:
16          canvas.destroy()
17
18      canvas = Canvas(window, width=width, height=height)
19      paper = PhotoImage(width=width, height=height)
20      canvas.create_image((width / 2, height / 2), image=paper, state="normal")
21      rgbString = ""
22      rgbImage = img.convert('RGB')
23      for i in range(0, height):
24          tmpString = ""
25          for k in range(0, width):
26              r, g, b = rgbImage.getpixel((k, i))
27              tmpString += "#%02x%02x%02x " % (r, g, b)   # x 뒤에 한 칸 공백
28          rgbString += "{" + tmpString + "} "             # } 뒤에 한 칸 공백
29      paper.put(rgbString)
30      canvas.pack(padx=20, pady=(20, 10))
```

이미지를 화면에 표시하는 함수

14: 창 크기 설정: 인자로 받아온 width와 height를 기반으로 창 크기 설정

15-16: canvas가 이미 존재한다면 제거

18-20: 캔버스와 이미지 객체 생성

21-28: RGB 문자열 생성: convert('RGB')를 사용해 img를 RGB 형식으로 변환 후 각 픽셀의 값을 for문과 getpixel()을 사용해 문자열로 변환

29-30: 생성된 RGB 문자열을 이용하여 이미지 생성 후 캔버스에 배치. 캔버스는 창에 배치. padx와 pady로 여백 설정

2.2 파일

2.2.1 파일 열기

```
33  def func_open():
34      global window, canvas, paper, photo, photo2, oriX, oriY
35      readFp = askopenfilename(parent=window, filetypes=(
36          ("jpg 파일", "*.jpg"), ("png 파일", "*.png"), ("gif 파일", "*.gif"),
37          ("jpeg 파일", "*.jpeg"), ("bmp 파일", "*.bmp"), ("tif 파일", "*.tif"), ("모든 파일", "*.*")))
38      photo = Image.open(readFp).convert('RGB')
39      oriX = photo.width
40      oriY = photo.height
41
42      photo2 = photo.copy()
43      newX = photo2.width
44      newY = photo2.height
45      displayImage(photo2, newX, newY)
```

이미지 파일을 열고 화면에 표시하는 함수

35-37: askopenfilename() 함수를 사용해 파일 읽음. jpg, png, gif, jpeg, bmp, tif 형식의 파일 선택 가능

38-40: 읽어온 파일을 Image.open() 함수로 열고, convert('RGB")로 RGB 형식으로 변환 후 이미지의 너비와 높이 정보 저장

42-44: 이미지와 이미지 정보 복사

45: 복사한 이미지를 창에 표시

2.2.2 파일 저장

```
48  def func_save():
49      global window, canvas, paper, photo, photo2, oriX, oriY
50
51      if photo2 == None:
52          return
53      saveFp = asksaveasfile(parent=window, mode="w", defaultextension=".jpg", filetypes=(
54          ("JPG 파일", "*.jpg; *jpeg"), ("모든 파일", "*.*")))
55
56      photo2.save(saveFp.name)
```

화면에 표시중인 이미지를 저장하는 함수

53: asksaveasfile() 함수를 사용해 이미지 저장 경로 선택. 확장자는 jpg 형식으로 저장

56: save() 함수를 사용해 photho2 이미지를 저장. saveFp는 경로와 이름을 나타낸다.

2.3 이미지 편집

```
64  def func_zoomin():
65      global window, canvas, paper, photo, photo2, oriX, oriY, scale_window
66
67      def get_scale_value():
68          global photo2, oriX, oriY
69          value = scale_value.get()
70          if value == 0.0:
71              value = scale.get()
72          photo2 = photo2.resize((int(oriX * value), int(oriY * value)))
73          newX = photo2.width
74          newY = photo2.height
75          displayImage(photo2, newX, newY)
76          scale_window.destroy()
77
78      scale_window = Tk()
79      scale_window.title("확대 배율 선택")
80      scale_window.geometry("400x150")
81
82      scale_value = DoubleVar()
83      scale = Scale(scale_window, variable=scale_value, from_=1, to=4,
84                    resolution=0.01, orient="horizontal", length=400, tickinterval=0.5)
85      scale.pack(padx=15, pady=15)
86
87      btn = Button(scale_window, text="확인",
88                   command=get_scale_value, width=8, height=1)
89      btn.pack(padx=5, pady=5)
90
91      window.mainloop()
```

이미지를 확대하는 함수

78-91: 확대 배율을 선택하는 창 열기. 사용자는 수치 조정 바를 이용하여 배율 설정. 설정한 값은 double 자료형으로 변수명 scale_value에 저장. 값은 1부터 4까지 0.01단위로 설정 가능.

67-76: [확인] 버튼을 클릭하면 시행되는 함수. get() 함수를 사용하여 사용자가 선택한 값 불러와 변수명 value에 저장. 그 후 resize()함수를 이용하여 기존 너비와 높이에 value를 곱한 값으로 photho2의 크기 변경. 변경한 photo2를 displayImage() 함수를 이용하여 화면에 표시.

이 외에도 축소, 반전, 회전, 밝게, 블러, 부드럽게 등 여러 기능이 있다. 기능들 대부분이 이와 비슷하거나 간단하기 때문에 생략

3 데이터베이스 구축

3.1 저장

```
354  def insert_image_to_db():
355      global name_window, entry_name
356
357      name_window = Tk()
358      name_window.title("이름 입력")
359      name_window.geometry("200x110")
360
361      label_name = Label(name_window, text="이미지 이름 입력")
362      label_name.pack(padx=5, pady=8)
363
364      entry_name = Entry(name_window)
365      entry_name.pack(padx=10, pady=(3, 10))
366
367      btn_submit = Button(name_window, text="확인",
368                          command=input_name, width=8, height=1)
369      btn_submit.pack(padx=5, pady=5)
370
371      name_window.mainloop()
```

이미지의 이름을 입력받는 함수

364-365: 이름을 입력받는 Entry 위젯 생성

367-369: input_name 함수를 실행시키는 버튼

```
341  def input_name():
342      global entry_name, name_window
343
344      name = entry_name.get().strip()
345
346      if name == "":
347          messagebox.showwarning("경고", "이미지 이름을 입력하세요.")
348          return
349
350      insert_database(name)
351      name_window.destroy()
```

이름 입력이 빈 문자열인지 확인하는 함수

빈 문자열이 입력되었다면 경고 메시지 출력 후 종료. 아니라면 insert_database 함수에 입력한 이름을 삽입하여 실행

```
302  def insert_database(name):
303      global path, photo, photo2, idCount
304      path = os.path.expanduser('~/imgDB').replace('\\', '/')
305
306      con = sqlite3.connect(path)
307      cur = con.cursor()
308
309      cur.execute(
310          "CREATE TABLE IF NOT EXISTS imageDB (id INTEGER PRIMARY KEY, name TEXT, extension TEXT, image BLOB)")
311
312      # 이미지 이름 중복 체크
313      cur.execute("SELECT name FROM imageDB WHERE name=?", (name,))
314      existing_name = cur.fetchone()
315
316      if existing_name:
317          result = messagebox.askquestion(
318              "중복 확인", "이미 동일한 이름이 존재합니다. 기존 데이터를 덮어쓰시겠습니까?")
319          if result == 'no':
320              con.close()
321              return
322
323      extension = photo.format
324
325      image_data = io.BytesIO()
326      photo2.save(image_data, format='JPEG')
327      image_data = image_data.getvalue()
328
329      if existing_name:
330          cur.execute("UPDATE imageDB SET extension=?, image=? WHERE name=?",
331                      (extension, image_data, name))
332      else:
333          cur.execute("INSERT INTO imageDB (name, extension, image) VALUES (?, ?, ?)",
334                      (name, extension, image_data))
335
336      messagebox.showinfo("저장 완료", "이미지 저장이 완료되었습니다.")
337      con.commit()
338      con.close()
```

이미지를 데이터베이스에 저장하는 함수

304: os.path.expanduser() 함수를 사용하여 경로를 Documents/imgDB로 설정

306-307: sqlite3를 사용해 데이터베이스 연결

309-310: imageDB라는 이름의 테이블 생성. 변수는 id, name, extension, image가 있다.

313-321: 이름 중복 체크. SELECT를 사용하여 imageDB 테이블에 name 열을 조회 후 WHERE을 사용하여 name 값이 변수 name과 일치하는 행을 찾는다. 그 후 fetchone()을 사용하여 찾은 행들 중 첫 번째 행을 existing_name 변수에 저장. 만약 값이 있다면 중복되는 이름이 있다는 뜻이므로 덮어쓰기 확인 창을 생성. [no]를 클릭한 경우 함수 종료

323: format을 사용해 원본 파일의 확장자를 가져온다

325-327: 메모리 내에서 바이트 데이터를 읽고 쓰기 위한 임시 버퍼를 생성. 편집한 이미지(photo2)를 JPEG 형식으로 버퍼에 저장. getvalue()를 사용하여 버퍼의 데이터를 바이트로 가져온다.

329-334: 이미지 이름이 기존에 존재하는 경우(중복) UPDATE를 사용하여 기존에 있던 데이터에 업데이트하고, 이름이 기존에 존재하지 않는 경우에 INSERT를 사용하여 새로운 레코드를 삽입한다.

336-338: 저장 완료 창을 띄운 뒤 변경사항을 커밋하고 DB 연결을 종료

3.2 조회

```
374  def load_database():
375      global path, photo, photo2, oriX, oriY, window, canvas, name_window, name_listbox, load_button
376
377      path = os.path.expanduser('~/imgDB').replace('\\', '/')
378
379      con = sqlite3.connect(path)
380      cur = con.cursor()
381
382      cur.execute("SELECT name FROM imageDB")
383      rows = cur.fetchall()
384      con.close()
385
386      if rows:
387          name_window = Tk()
388          name_window.title("이미지 선택")
389          name_window.geometry("200x370")
390
391          label_name = Label(name_window, text="이미지 선택")
392          label_name.pack(pady=5)
393
394          name_listbox = Listbox(name_window, height=15)
395          for row in rows:
396              name_listbox.insert(END, row[0])
397          name_listbox.pack(pady=(0, 10))
398
399          load_button = Button(name_window, text="로드",
400                               command=load_image, width=8, height=1)
401          load_button.pack(padx=10, pady=5)
402
403          delete_button = Button(name_window, text="삭제",
404                                 command=delete_DB, width=8, height=1)
405          delete_button.pack(pady=5)
406
407          name_window.mainloop()
408      else:
409          messagebox.showinfo("로드 실패", "로드할 이미지가 없습니다.")
```

데이터베이스에서 이미지 이름들을 가져와 이미지를 선택할 수 있는 창을 생성하는 함수

377-380: path 경로의 데이터베이스에 연결

382-384: SELECT를 사용하여 imageDB의 모든 이름을 가져오고 연결 종료

386-407: 이미지가 있는 경우 새 창 생성.

395-396 라인의 for문을 통해 리스트박스에 이미지 이름을 추가. insert() 함수를 사용하여 리스트박스 마지막(END)에 현재 이미지 이름(row[0]) 값을 추가.

408-409: DB에 저장된 이미지가 없다면 로드 실패 메시지 표시

3.3 로드

```
412  def load_image():
413      global name_window, name_listbox, load_button, path, photo, photo2, oriX, oriY, window, canvas
414
415      selected_name = name_listbox.get(ANCHOR)
416      path = os.path.expanduser('~/imgDB').replace('\\', '/')
417
418      con = sqlite3.connect(path)
419      cur = con.cursor()
420
421      cur.execute("SELECT image FROM imageDB WHERE name=?", (selected_name,))
422      row = cur.fetchone()
423      con.close()
424
425      if row:
426          messagebox.showinfo("로드 완료", "이미지 로드가 완료되었습니다.")
427          photo = Image.open(io.BytesIO(row[0])).convert('RGB')
428          oriX = photo.width
429          oriY = photo.height
430
431          photo2 = photo.copy()
432          newX = photo2.width
433          newY = photo2.height
434          displayImage(photo2, newX, newY)
435          name_window.destroy()
436      else:
437          messagebox.showinfo("로드 실패", "로드할 이미지가 없습니다.")
```

선택한 이미지를 데이터베이스에서 로드하여 화면에 표시하는 함수

415: 사용자가 리스트 박스에서 선택한 값을 가져온다.

416-419: 지정된 경로의 데이터베이스에 연결

421-423: SELECT를 사용하여 선택한 이름의 이미지 데이터를 가져오고 데이터베이스 연결 해제

425-435: 데이터를 가져오는데 성공했을 경우. 로드 완료 메시지 표시. io.BytesIO를 사용하여 row[0]에 저장된 이미지 데이터를 이미지 객체로 변환하고, convert()를 사용하여 RGB 형식으로 변환. 가져온 이미지를 원본 이미지 변수(photo)에 저장. photo2 변수에 원본 이미지를 복사하고 displayImage()를 사용하여 복사한 이미지를 창에 표시

436-437: 가져온 데이터가 없는 경우 로드 실패 메시지

3.4 삭제

```
440  def delete_DB():
441      global name_window, name_listbox, load_button, path, photo, photo2, oriX, oriY, window, canvas
442
443      selected_name = name_listbox.get(ANCHOR)
444      path = os.path.expanduser('~/imgDB').replace('\\', '/')
445
446      con = sqlite3.connect(path)
447      cur = con.cursor()
448
449      cur.execute("DELETE FROM imageDB WHERE name=?", (selected_name,))
450      con.commit()
451      con.close()
452
453      messagebox.showinfo("삭제 완료", "데이터 삭제가 완료되었습니다.")
454      name_window.destroy()
455      load_database()
```

선택한 이미지를 데이터베이스에서 삭제하는 함수

443: 사용자가 리스트 박스에서 선택한 값을 가져온다.

444-447: 지정된 경로의 데이터베이스에 연결

449: DELETE를 사용하여 선택한 이미지 이름의 데이터를 삭제

450-451: 데이터베이스에 변경사항을 적용 후 연결 해제.

453-455 삭제 완료 메시지 표시 후 기존 이름 선택 창을 닫고 load_database() 함수를 사용하여 리스트 박스를 업데이트

4 참고자료

파이썬 for Begginer(지정도서), 구글링, chatGPT

5 결론

이 프로그램은 사용자가 이미지 편집 기능을 쉽고 편리하게 수행할 수 있는 기능을 제공합니다. tkinter를 통해 제공되는 사용자 친화적인 인터페이스는 이미지 선택 및 처리 과정을 간단하게 만들어줍니다. 또한, SQLite 데이터베이스를 활용하여 사용자가 작업한 이미지를 보관하고 추후에도 사용할 수 있도록 합니다. 이를 통해 사용자는 이미지 처리 작업을 보다 효율적으로 수행할 수 있으며, 필요한 경우 작업 이력을 확인하고 수정할 수 있습니다.