```
 1 import pygame
 2
 3 class Text(object):
 4     """description of class"""
 5
 6     def Write(surface,pos,text,fontSize,fontFamily,
   fontColor,centered = False):
 7         outputText = pygame.font.Font(fontFamily,
   fontSize)
 8         textSurf = outputText.render(text,True,
   fontColor)
 9         textRect = outputText.render(text,True,
   fontColor).get_rect()
10         trect = textRect
11         if centered:
12             textRect.center = (pos[0],pos[1])
13         else:
14             textRect = (pos[0],pos[1])
15         surface.blit(textSurf,textRect)
16
17         return (pos[0],pos[1],trect[2],trect[3])
18
19
20     def WriteMultiline(surface, text, pos, font,
   color=(0,0,0)):
21         words = [word.split(' ') for word in text.
   splitlines()]
22         space = font.size(' ')[0]
23         max_width, max_height = surface.get_size()
24         x, y = pos
25         for line in words:
26             for word in line:
27                 word_surface = font.render(word, 0,
   color)
28                 word_width, word_height =
   word_surface.get_size()
29                 if x + word_width >= max_width:
30                     x = pos[0]
31                     y += word_height
32                 surface.blit(word_surface, (x, y))
33                 x += word_width + space
34             x = pos[0]
35             y += word_height
36
```

```
37          return (pos[0],pos[1],word_width,word_height)
38
39
```

```
 1 # Importing Modules
 2 import pygame
 3 from pygame.locals import *
 4 import time
 5 import random
 6 import sys
 7 import math
 8
 9 from DexInfo import DexInfo
10 from DexMenu import DexMenu
11 from DexHome import DexHome
12 from DexSearch import DexSearch
13
14 # PyGame Initialisation
15 pygame.init()
16 clock = pygame.time.Clock()
17
18 # Window and Surface Initialisation
19 displayWidth = 800
20 displayHeight = 480
21
22 flags = FULLSCREEN | DOUBLEBUF
23
24 try:
25     if os.uname()[1] == 'raspberrypi':
26         mainSurface = pygame.display.set_mode((0,0),
   flags)
27         pygame.mouse.set_cursor((8,8),(0,0),(0,0,0,0,
   0,0,0,0),(0,0,0,0,0,0,0,0))
28     else:
29         mainSurface = pygame.display.set_mode((
   displayWidth,displayHeight))
30         pygame.mouse.set_visible(True)
31 except:
32     mainSurface = pygame.display.set_mode((
   displayWidth,displayHeight))
33     pygame.mouse.set_visible(True)
34
35
36 while True:
37
38     # Event Processing
39     for event in pygame.event.get():
40         if event.type == pygame.QUIT:
```

```
41                pygame.quit()
42                sys.exit()
43
44        # Keypress Processing
45        keys = pygame.key.get_pressed()
46        if keys[pygame.K_q] != 0:
47            pygame.quit()
48            sys.exit()
49
50        DexHome.Show()
51
52        pygame.display.update()
53        clock.tick(1)
54
55
56
57
58
59
60
61
```

```
 1 import pygame
 2 import time
 3 from pygame import gfxdraw
 4 from CText import Text
 5 from CDrawing import Draw
 6 from threading import Thread
 7
 8 class Button:
 9
10     sleepThread = Thread()
11     idleColor = (0,0,0)
12     hoverColor = (40,40,40)
13     disabledColor = (60,60,60)
14     fontColor = (255,255,255)
15     borderColor = (255,255,255)
16     clickCooldown = 0.2
17
18     fontFamily = "default"
19
20     class RoundRect:
21
22         def __init__(self, surface, rect, radius,
   text, fontSize, borderWidth = None, clickAction =
   None, clickParameters = None, releaseAction = None,
   releaseParameters = None,triggerMarginTB = 0,
   triggerMarginLR = 0):
23             self.surface = surface
24             self.rect = rect
25             self.radius = radius
26             self.text = text
27             self.fontSize = fontSize
28             self.borderWidth = borderWidth
29             self.triggerMarginTB = triggerMarginTB
30             self.triggerMarginLR = triggerMarginLR
31
32             self.idleColor = Button.idleColor
33             self.hoverColor = Button.hoverColor
34             self.disabledColor = Button.disabledColor
35             self.fontColor = Button.fontColor
36             self.borderColor = Button.borderColor
37
38             self.fontFamily = Button.fontFamily
39
40             self.clickAction = clickAction
```

```
41              self.clickParameters = clickParameters
42              self.releaseAction = releaseAction
43              self.releaseParameters =
   releaseParameters
44
45              self.lastClickState = 0
46
47          def Show(self, enabled = True, disabled =
   False):
48              mouse = pygame.mouse.get_pos()
49              click = pygame.mouse.get_pressed()
50
51              if not disabled:
52                  if enabled:
53                      if self.rect[0] - self.
   triggerMarginLR < mouse[0] < self.rect[0] + self.rect
   [2] + self.triggerMarginLR and self.rect[1] - self.
   triggerMarginTB < mouse[1] < self.rect[1] + self.rect
   [3] + self.triggerMarginTB:
54
55                          Draw.RoundRect(self.surface,
   self.hoverColor,self.rect,self.radius,self.
   borderWidth,self.borderColor)
56
57                          if not Button.sleepThread.
   isAlive():
58                              if click[0] == 1:
59                                  if self.clickAction
    != None:
60                                      if self.
   clickParameters != None: self.clickAction(self.
   clickParameters)
61                                      else: self.
   clickAction()
62
63                                  Button.sleepThread =
   Thread(target = time.sleep, args = (Button.
   clickCooldown,))
64                                  Button.sleepThread.
   start()
65
66                              if click[0] == 0 and self
   .lastClickState == 1:
67                                  if self.releaseAction
```

```
 != None:
                                        if self.releaseParameters != None: self.releaseAction(self.releaseParameters)
                                        else: self.releaseAction()
                        else: Draw.RoundRect(self.surface,self.idleColor,self.rect,self.radius,self.borderWidth,self.borderColor)
                    else: Draw.RoundRect(self.surface,self.idleColor,self.rect,self.radius,self.borderWidth,self.borderColor)
                else: Draw.RoundRect(self.surface,self.disabledColor,self.rect,self.radius,self.borderWidth,self.borderColor)

                Text.Write(self.surface,(self.rect[0]+self.rect[2]/2,self.rect[1]+self.rect[3]/2),self.text,self.fontSize,self.fontFamily,self.fontColor,True)

                self.lastClickState = click[0]

                if self.borderWidth != None: return (self.rect[0]-self.borderWidth,self.rect[1]-self.borderWidth,self.rect[2]+2*self.borderWidth,self.rect[3]+2*self.borderWidth)
                else: return self.rect
```

```
 1 # Importing Modules
 2 import pygame
 3 from pygame import gfxdraw
 4 from pygame.locals import *
 5 from threading import Thread
 6 import time
 7 import random
 8 import sys
 9 import sqlite3
10 import os
11 import math
12
13 from CButton import Button
14 from SpriteManager import Sprite
15 from CDrawing import Draw
16 from CText import Text
17
18 import CText
19
20 from DexMenu import DexMenu
21
22
23 class DexHome:
24
25 ##########################################################
   ####################################
26 #   PROTECTED VARIABLES

             #
27 ##########################################################
   ####################################
28
29     conn = sqlite3.connect('pokemon.db')
30     conn.row_factory = sqlite3.Row
31     c = conn.cursor()
32
33     thread = Thread()
34     sleepThread = Thread()
35
36     running = True
37     reload = True
38     selectionMade = False
39     selectedOption = None
40
```

```
41      selectedScreen = 1
42
43  ##################################################################################
44  #   FUNCTIONS
45  ##################################################################################
46
47
48  ##################################################################################
49  #   TOGGLE FUNCTIONS
50  ##################################################################################
51
52
53
54  ##################################################################################
55  ##################################################################################
56  #   MAIN START
57  ##################################################################################
58  ##################################################################################
59
60      def Show():
61
62  ##################################################################################
63  #   INITIALISATION AND SETUP
64  ##################################################################################
65
66          # PyGame Initialisation
```

```
67          clock = pygame.time.Clock()
68
69          # Window and Surface Initialisation
70          displayWidth = 800
71          displayHeight = 480
72
73          idleCtr = 0
74
75          flags = FULLSCREEN | DOUBLEBUF
76
77          try:
78              if os.uname()[1] == 'raspberrypi':
79                  mainSurface = pygame.display.
    set_mode((0,0),flags)
80                  pygame.mouse.set_cursor((8,8),(0,0
    ),(0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,0))
81              else:
82                  mainSurface = pygame.display.
    set_mode((displayWidth,displayHeight))
83                  pygame.mouse.set_visible(True)
84          except:
85              mainSurface = pygame.display.set_mode((
    displayWidth,displayHeight))
86              pygame.mouse.set_visible(True)
87
88          pygame.event.set_allowed([QUIT, KEYDOWN,
    KEYUP])
89
90  ##################################################
    ######################################
91  #   TEMA ENTRADA

                        #
92  ##################################################
    ######################################
93
94
95  ##################################################
    ######################################
96  #   DEFINICOES DO MENU INICIAL

          #
97  ##################################################
    ######################################
```

```
 98
 99          fadeOutCtr = 0
100
101 #espessura bordas
102          borderWidth = 5
103 #cor linhas iniciais
104          borderColor = (255,0,0)
105 #cor da caixa de menu
106          infillColor = (20,20,20)
107 #fonte tela inicial
108          fontColor = (255,255,255)
109 #fundo tela inicial
110          backColor = (0,0,0)
111 #angulo de raio
112          radius = 20
113 #distancia dos circulos
114          sectionPadding = 12
115 #correcao de borda
116          borderCorrection = 1
117 #tamanho da fonte
118          fontSize = 35
119 #tamanho menu circular
120          centerRadius = 130
121  # Tempo para uso
122          repressCooldown = 10
123
124 ##################################################
    ######################################
125 #   LOADING LOOP

                        #
126 ##################################################
    ######################################
127
128          while DexHome.running:
129
130              DexHome.reload = False
131
132 ##################################################
    ######################################
133 #   RUNNING LOOP

                        #
134 ##################################################
```

```
134 ########################################
135
136             while not DexHome.reload:
137
138                 # Event Processing
139                 for event in pygame.event.get():
140                     if event.type == pygame.QUIT:
141                         pygame.quit()
142                         sys.exit()
143
144                 # Keypress Processing
145                 keys = pygame.key.get_pressed()
146                 if keys[pygame.K_q] != 0:
147                     pygame.quit()
148                     sys.exit()
149
150                 mouse = pygame.mouse.get_pos()
151                 click = pygame.mouse.get_pressed()
152
153
154                 #cor fundo principal
155                 mainSurface.fill((0,0,0))
156
157 ##################################################
    ######################################
158 #   Tela 1

                                #
159 ##################################################
    ######################################
160
161                 if DexHome.selectedScreen == 1:
162
163                     boxOffset1 = 300
164                     boxOffset2 = 360
165                     boxOffset3 = 420
166                     boxOffset4 = -420
167                     boxOffset5 = -360
168                     boxOffset6 = -300
169
170                     boxWidth = int(266.66 - 2*10)
171                     boxHeight = 240-2  *10
172
173                     btnColorCenter = infillColor
```

```
174                         btnColorTopLeft = infillColor
175                         btnColorTopRight = infillColor
176                         btnColorBottomLeft = infillColor
177                         btnColorBottomRight =
    infillColor
178 
179                         btnFontCenter = fontColor
180                         btnFontTopLeft = fontColor
181                         btnFontTopRight = fontColor
182                         btnFontBottomLeft = fontColor
183                         btnFontBottomRight = fontColor
184 
185 
186                         if repressCooldown <= 0 and not
    DexHome.selectionMade:
187 
188                             # Button functionality
189                             if 400-centerRadius < mouse[
    0] < 400+centerRadius and 240-centerRadius < mouse[1
    ] < 240+centerRadius:
190                                 btnColorCenter = (255,
    255,255)
191                                 btnFontCenter = (255,0,0
    )
192                                 if click[0] == 1:
193                                     DexHome.
    selectionMade = True
194                                     DexHome.
    selectedOption = 1
195 
196                             if (0 < mouse[0] < 400 and 0
     < mouse[1] < 240-centerRadius) or (0 < mouse[0] <
    400-centerRadius and 0 < mouse[1] < 240):
197                                 btnColorTopLeft = (255,
    255,255)
198                                 btnFontTopLeft = (255,0,
    0)
199                                 if click[0] == 1:
200                                     DexHome.
    selectionMade = True
201                                     DexHome.
    selectedOption = 2
202 
203                             if (400 < mouse[0] < 800 and
```

```
203  0 < mouse[1] < 240-centerRadius) or (400+
    centerRadius < mouse[0] < 800 and 0 < mouse[1] < 240
    ):
204                                 btnColorTopRight = (255,
    255,255)
205                                 btnFontTopRight = (255,0
    ,0)
206                                 if click[0] == 1:
207                                     DexHome.
    selectionMade = True
208                                     DexHome.
    selectedOption = 3
209
210                             if (0 < mouse[0] < 400 and
    240+centerRadius < mouse[1] < 480) or (0 < mouse[0
    ] < 400-centerRadius and 240 < mouse[1] < 480):
211                                 btnColorBottomLeft = (
    255,255,255)
212                                 btnFontBottomLeft = (255
    ,0,0)
213                                 if click[0] == 1:
214                                     DexHome.
    selectionMade = True
215                                     DexHome.
    selectedOption = 4
216
217                             if (400 < mouse[0] < 800 and
     240+centerRadius < mouse[1] < 480) or (400+
    centerRadius < mouse[0] < 800 and 240 < mouse[1] <
    480):
218                                 btnColorBottomRight = (
    255,255,255)
219                                 btnFontBottomRight = (
    255,0,0)
220                                 if click[0] == 1:
221                                     DexHome.
    selectionMade = True
222                                     DexHome.
    selectedOption = 5
223
224
225
226
227
```

```
228                         # Top Left Segment
229                         # Edge Circles
230                         Draw.AAFilledCircle(mainSurface,
    sectionPadding+radius,sectionPadding+radius,radius,
    btnColorTopLeft,borderColor,borderWidth)
231                         Draw.AAFilledCircle(mainSurface,
    400-sectionPadding-radius,sectionPadding+radius,
    radius,btnColorTopLeft,borderColor,borderWidth)
232                         Draw.AAFilledCircle(mainSurface,
    sectionPadding+radius,240-sectionPadding-radius,
    radius,btnColorTopLeft,borderColor,borderWidth)
233
234                         # Outer Connections
235                         pygame.draw.line(mainSurface,
    borderColor,(sectionPadding+radius,sectionPadding+
    borderCorrection),(400-sectionPadding-radius,
    sectionPadding+borderCorrection),borderWidth)
236                         pygame.draw.line(mainSurface,
    borderColor,(sectionPadding+borderCorrection,
    sectionPadding+radius),(sectionPadding+
    borderCorrection,240-sectionPadding-radius),
    borderWidth)
237                         pygame.draw.line(mainSurface,
    borderColor,(sectionPadding+radius,240-
    sectionPadding-borderCorrection),(800-sectionPadding
    -radius,240-sectionPadding-borderCorrection),
    borderWidth)
238                         pygame.draw.line(mainSurface,
    borderColor,(400-sectionPadding-borderCorrection,
    sectionPadding+radius+borderCorrection),(400-
    sectionPadding-borderCorrection,480-sectionPadding-
    radius),borderWidth)
239
240                         # Infills
241                         pygame.draw.rect(mainSurface,
    btnColorTopLeft,(sectionPadding+borderWidth,
    sectionPadding+radius,400-2*sectionPadding-2*
    borderWidth+borderCorrection,240-4*sectionPadding-2*
    borderWidth))
242                         pygame.draw.rect(mainSurface,
    btnColorTopLeft,(sectionPadding+radius,
    sectionPadding+borderWidth,400-radius-2*
    sectionPadding-4*borderWidth+borderCorrection,240-2*
    sectionPadding-2*borderWidth+borderCorrection))
```

```
243
244                     # Top Right Segment
245                     # Edge Circles
246                     Draw.AAFilledCircle(mainSurface,
    400+sectionPadding+radius,sectionPadding+radius,
    radius,btnColorTopRight,borderColor,borderWidth)
247                     Draw.AAFilledCircle(mainSurface,
    800-sectionPadding-radius,sectionPadding+radius,
    radius,btnColorTopRight,borderColor,borderWidth)
248                     Draw.AAFilledCircle(mainSurface,
    800-sectionPadding-radius,240-sectionPadding-radius,
    radius,btnColorTopRight,borderColor,borderWidth)
249
250                     # Outer Connections
251                     pygame.draw.line(mainSurface,
    borderColor,(400+sectionPadding+radius,
    sectionPadding+borderCorrection),(800-sectionPadding
    -radius,sectionPadding+borderCorrection),borderWidth
    )
252                     pygame.draw.line(mainSurface,
    borderColor,(sectionPadding+borderCorrection,240+
    sectionPadding+radius),(sectionPadding+
    borderCorrection,480-sectionPadding-radius),
    borderWidth)
253                     pygame.draw.line(mainSurface,
    borderColor,(400+sectionPadding+borderCorrection,
    sectionPadding+radius+borderCorrection),(400+
    sectionPadding+borderCorrection,480-sectionPadding-
    radius),borderWidth)
254
255                     # Infills
256                     pygame.draw.rect(mainSurface,
    btnColorTopRight,(400+sectionPadding+borderWidth,
    sectionPadding+radius,400-2*sectionPadding-2*
    borderWidth+borderCorrection,240-4*sectionPadding-2*
    borderWidth))
257                     pygame.draw.rect(mainSurface,
    btnColorTopRight,(400+sectionPadding+radius,
    sectionPadding+borderWidth,400-radius-2*
    sectionPadding-4*borderWidth+borderCorrection,240-2*
    sectionPadding-2*borderWidth+borderCorrection))
258
259
260                     # Bottom Left Segment
```

```
261                         # Edge Circles
262                         Draw.AAFilledCircle(mainSurface,
    sectionPadding+radius,240+sectionPadding+radius,
    radius,btnColorBottomLeft,borderColor,borderWidth)
263                         Draw.AAFilledCircle(mainSurface,
    400-sectionPadding-radius,480-sectionPadding-radius,
    radius,btnColorBottomLeft,borderColor,borderWidth)
264                         Draw.AAFilledCircle(mainSurface,
    sectionPadding+radius,480-sectionPadding-radius,
    radius,btnColorBottomLeft,borderColor,borderWidth)
265
266                         # Outer Connections
267                         pygame.draw.line(mainSurface,
    borderColor,(sectionPadding+radius,480-
    sectionPadding-borderCorrection),(400-sectionPadding
    -radius,480-sectionPadding-borderCorrection),
    borderWidth)
268                         pygame.draw.line(mainSurface,
    borderColor,(800-sectionPadding-borderCorrection,
    sectionPadding+radius),(800-sectionPadding-
    borderCorrection,240-sectionPadding-radius),
    borderWidth)
269                         pygame.draw.line(mainSurface,
    borderColor,(sectionPadding+radius,240+
    sectionPadding+borderCorrection),(800-sectionPadding
    -radius,240+sectionPadding+borderCorrection),
    borderWidth)
270
271                         # Infills
272                         pygame.draw.rect(mainSurface,
    btnColorBottomLeft,(sectionPadding+borderWidth,240+
    sectionPadding+radius,400-2*sectionPadding-2*
    borderWidth+borderCorrection,240-4*sectionPadding-2*
    borderWidth))
273                         pygame.draw.rect(mainSurface,
    btnColorBottomLeft,(sectionPadding+radius,240+
    sectionPadding+borderWidth,400-radius-2*
    sectionPadding-4*borderWidth+borderCorrection,240-2*
    sectionPadding-2*borderWidth+borderCorrection))
274
275                         # Bottom Right Segment
276                         # Edge Circles
277                         Draw.AAFilledCircle(mainSurface,
    400+sectionPadding+radius,480-sectionPadding-radius,
```

```
277 radius,btnColorBottomRight,borderColor,borderWidth)
278                         Draw.AAFilledCircle(mainSurface,
    800-sectionPadding-radius,480-sectionPadding-radius,
    radius,btnColorBottomRight,borderColor,borderWidth)
279                         Draw.AAFilledCircle(mainSurface,
    800-sectionPadding-radius,240+sectionPadding+radius,
    radius,btnColorBottomRight,borderColor,borderWidth)
280 
281                         # Outer Connections
282                         pygame.draw.line(mainSurface,
    borderColor,(400+sectionPadding+radius,480-
    sectionPadding-borderCorrection),(800-sectionPadding
    -radius,480-sectionPadding-borderCorrection),
    borderWidth)
283                         pygame.draw.line(mainSurface,
    borderColor,(800-sectionPadding-borderCorrection,240
    +sectionPadding+radius),(800-sectionPadding-
    borderCorrection,480-sectionPadding-radius),
    borderWidth)
284 
285                         # Infills
286                         pygame.draw.rect(mainSurface,
    btnColorBottomRight,(400+sectionPadding+borderWidth,
    240+sectionPadding+radius,400-2*sectionPadding-2*
    borderWidth+borderCorrection,240-4*sectionPadding-2*
    borderWidth))
287                         pygame.draw.rect(mainSurface,
    btnColorBottomRight,(400+sectionPadding+radius,240+
    sectionPadding+borderWidth,400-radius-2*
    sectionPadding-4*borderWidth+borderCorrection,240-2*
    sectionPadding-2*borderWidth+borderCorrection))
288 
289                         # Outer Segments Large Radius
290                         Draw.AAFilledCircle(mainSurface,
    400,240,centerRadius+2*sectionPadding,backColor,
    borderColor,borderWidth)
291                         pygame.draw.rect(mainSurface,
    backColor,(400-sectionPadding+borderCorrection,0,2*
    sectionPadding-borderCorrection,480))
292                         pygame.draw.rect(mainSurface,
    backColor,(0,240-sectionPadding+borderCorrection,800
    ,2*sectionPadding-borderCorrection))
293 
294                         # Center segment
```

```
295                     Draw.AAFilledCircle(mainSurface,
    400,240,centerRadius,btnColorCenter,borderColor,
    borderWidth)
296
297
298
    ##########################################################
    ##############################
299                     # Topicos do menu

            #
300
    ##########################################################
    ##############################
301
302
303                     Text.Write(mainSurface,(400,240
    ),"Pokedex",fontSize,"joy.otf",btnFontCenter,True)
304
305                     Text.Write(mainSurface,(170,120
    ),"---",fontSize,"joy.otf",btnFontTopLeft,True)
306
307                     Text.Write(mainSurface,(645,120
    ),"---",fontSize,"joy.otf",btnFontTopRight,True)
308
309
310                     Text.Write(mainSurface,(170,370
    ),"---",fontSize,"joy.otf",btnFontBottomLeft,True)
311
312                     Text.Write(mainSurface,(645,370
    ),"Mais >",fontSize,"joy.otf",btnFontBottomRight,
    True)
313
314                     # fade entre telas
315                     if DexHome.selectionMade:
316                         if fadeOutCtr <= 1:
317                             sectionPadding -= 1
318                             centerRadius += 1
319                         if fadeOutCtr > 1:
320                             sectionPadding += 1
321                             if centerRadius >= 2:
    centerRadius -= 3
322                             if fontSize > 0:
    fontSize -= 1
```

```
323                             if fadeOutCtr > 100:
324                                 fadeOutColor = (0,0,0)
325                                 Draw.AAFilledCircle(
    mainSurface,400,240,(fadeOutCtr-10)*40,fadeOutColor,
    fadeOutColor,3)
326                             if fadeOutCtr > 11:
327 
328                                 sectionPadding = 10
329                                 fontSize = 35
330                                 centerRadius = 130
331                                 fadeOutCtr = 0
332                                 DexHome.selectionMade=
    False
333 
334                                 if DexHome.
    selectedOption == 1: DexMenu.Show()
335                                 if DexHome.
    selectedOption == 2: pass
336                                 if DexHome.
    selectedOption == 3: pass
337                                 if DexHome.
    selectedOption == 4: pass
338                                 if DexHome.
    selectedOption == 5: DexHome.selectedScreen = 2
339 
340                                 repressCooldown = 10
341 
342                             fadeOutCtr += 1
343 
344                         if repressCooldown > 0:
    repressCooldown -= 1
345 
346 ##################################################
    ######################################
347 #   Mais >   Tela 2

                                #
348 ##################################################
    ######################################
349 
350 
351 
352                     if DexHome.selectedScreen == 2:
353 
```

```
354                         btnColorMore1 = infillColor
355                         btnColorMore2 = infillColor
356                         btnColorMore3 = infillColor
357                         btnColorMore4 = infillColor
358                         btnColorMore5 = infillColor
359                         btnColorMore6 = infillColor
360 
361                         btnFontMore1 = fontColor
362                         btnFontMore2 = fontColor
363                         btnFontMore3 = fontColor
364                         btnFontMore4 = fontColor
365                         btnFontMore5 = fontColor
366                         btnFontMore6 = fontColor
367 
368 
369                         while boxOffset1 > 0 or
    boxOffset2 > 0 or boxOffset3 > 0 or boxOffset4 > 0
    or boxOffset5 > 0 or boxOffset6 > 0:
370 
371                             mainSurface.fill((0,0,0))
372 
373                             Draw.RoundRect(mainSurface,
    btnColorMore1,(1*10+0*boxWidth,10-boxOffset1,
    boxWidth,boxHeight),radius,borderWidth,borderColor)
374                             Draw.RoundRect(mainSurface,
    btnColorMore2,(3*10+1*boxWidth,10-boxOffset2,
    boxWidth,boxHeight),radius,borderWidth,borderColor)
375                             Draw.RoundRect(mainSurface,
    btnColorMore3,(5*10+2*boxWidth,10-boxOffset3,
    boxWidth,boxHeight),radius,borderWidth,borderColor)
376 
377                             Draw.RoundRect(mainSurface,
    btnColorMore4,(1*10+0*boxWidth,3*10+boxHeight-
    boxOffset4,boxWidth,boxHeight),radius,borderWidth,
    borderColor)
378                             Draw.RoundRect(mainSurface,
    btnColorMore5,(3*10+1*boxWidth,3*10+boxHeight-
    boxOffset5,boxWidth,boxHeight),radius,borderWidth,
    borderColor)
379                             Draw.RoundRect(mainSurface,
    btnColorMore6,(5*10+2*boxWidth,3*10+boxHeight-
    boxOffset6,boxWidth,boxHeight),radius,borderWidth,
    borderColor)
380 
```

```
381                             if boxOffset1 > 0:
    boxOffset1 -= 20
382                             if boxOffset2 > 0:
    boxOffset2 -= 20
383                             if boxOffset3 > 0:
    boxOffset3 -= 20
384                             if boxOffset4 < 0:
    boxOffset4 += 20
385                             if boxOffset5 < 0:
    boxOffset5 += 20
386                             if boxOffset6 < 0:
    boxOffset6 += 20
387
388                             pygame.display.update()
389                             clock.tick(60)
390
391                         if repressCooldown <= 0:
392                             if 0 < mouse[0] < 266 and 0
     < mouse[1] < 240:
393                                 btnColorMore1 = (255,255
    ,255)
394                                 btnFontMore1 = (255,0,0)
395                                 if click[0] == 1:
396                                     DexHome.
    selectionMade = True
397                                     DexHome.
    selectedOption = 6
398
399                             elif 266 < mouse[0] < 533
    and 0 < mouse[1] < 240:
400                                 btnColorMore2 = (255,255
    ,255)
401                                 btnFontMore2 = (255,0,0)
402                                 if click[0] == 1:
403                                     DexHome.
    selectionMade = True
404                                     DexHome.
    selectedOption = 7
405
406                             elif 533 < mouse[0] < 800
    and 0 < mouse[1] < 240:
407                                 btnColorMore3 = (255,255
    ,255)
408                                 btnFontMore3 = (255,0,0)
```

```
409                                 if click[0] == 1:
410                                     DexHome.
    selectionMade = True
411                                     DexHome.
    selectedOption = 8
412
413                             elif 0 < mouse[0] < 266 and
    240 < mouse[1] < 480:
414                                 btnColorMore4 = (255,255
    ,255)
415                                 btnFontMore4 = (255,0,0)
416                                 if click[0] == 1:
417                                     DexHome.
    selectionMade = True
418                                     DexHome.
    selectedOption = 9
419
420                             elif 266 < mouse[0] < 533
    and 240 < mouse[1] < 480:
421                                 btnColorMore5 = (255,255
    ,255)
422                                 btnFontMore5 = (255,0,0)
423                                 if click[0] == 1:
424                                     DexHome.
    selectionMade = True
425                                     DexHome.
    selectedOption = 10
426
427                             elif 533 < mouse[0] < 800
    and 240 < mouse[1] < 480:
428                                 btnColorMore6 = (255,255
    ,255)
429                                 btnFontMore6 = (255,0,0)
430                                 if click[0] == 1:
431                                     DexHome.
    selectionMade = True
432                                     DexHome.
    selectedOption = 11
433
434
435                     Draw.RoundRect(mainSurface,
    btnColorMore1,(1*10+0*boxWidth,10,boxWidth,boxHeight
    ),radius,borderWidth,borderColor)
436                     Draw.RoundRect(mainSurface,
```

```
436 btnColorMore2,(3*10+1*boxWidth,10,boxWidth,boxHeight
    ),radius,borderWidth,borderColor)
437                         Draw.RoundRect(mainSurface,
    btnColorMore3,(5*10+2*boxWidth,10,boxWidth,boxHeight
    ),radius,borderWidth,borderColor)
438
439                         Draw.RoundRect(mainSurface,
    btnColorMore4,(1*10+0*boxWidth,3*10+boxHeight,
    boxWidth,boxHeight),radius,borderWidth,borderColor)
440                         Draw.RoundRect(mainSurface,
    btnColorMore5,(3*10+1*boxWidth,3*10+boxHeight,
    boxWidth,boxHeight),radius,borderWidth,borderColor)
441                         Draw.RoundRect(mainSurface,
    btnColorMore6,(5*10+2*boxWidth,3*10+boxHeight,
    boxWidth,boxHeight),radius,borderWidth,borderColor)
442
443                         Text.Write(mainSurface,(1*10+0*
    boxWidth+boxWidth/2,10+boxHeight/2),"< Voltar",
    fontSize,"joy.otf",btnFontMore1,True)
444
445                         Text.Write(mainSurface,(3*10+1*
    boxWidth+boxWidth/2,10+boxHeight/2),"---",fontSize,"
    joy.otf",btnFontMore2,True)
446
447                         Text.Write(mainSurface,(5*10+2*
    boxWidth+boxWidth/2,10+boxHeight/2-40),"Carta",
    fontSize,"joy.otf",btnFontMore3,True)
448                         Text.Write(mainSurface,(5*10+2*
    boxWidth+boxWidth/2,10+boxHeight/2),"de",fontSize,"
    joy.otf",btnFontMore3,True)
449                         Text.Write(mainSurface,(5*10+2*
    boxWidth+boxWidth/2,10+boxHeight/2+40),"Treinador",
    fontSize,"joy.otf",btnFontMore3,True)
450
451                         Text.Write(mainSurface,(1*10+0*
    boxWidth+boxWidth/2,3*10+boxHeight+boxHeight/2),
    "---",fontSize,"joy.otf",btnFontMore4,True)
452                         Text.Write(mainSurface,(3*10+1*
    boxWidth+boxWidth/2,3*10+boxHeight+boxHeight/2),"
    Atualizar",fontSize,"joy.otf",btnFontMore5,True)
453
454                         Text.Write(mainSurface,(5*10+2*
    boxWidth+boxWidth/2,3*10+boxHeight+boxHeight/2),"
    Sobre",fontSize,"joy.otf",btnFontMore6,True)
```

```
455
456
457                     if DexHome.selectionMade:
458                         if fadeOutCtr <= 3:
459                             boxWidth +=1
460                             boxHeight +=1
461                         if fadeOutCtr > 3:
462                             boxWidth -=5
463                             boxHeight -=5
464                             if fontSize > 0:
    fontSize -= 2
465                         if fadeOutCtr > 10:
466                             fadeOutColor = (0,0,0)
467                             Draw.AAFilledCircle(
    mainSurface,400,240,(fadeOutCtr-10)*40,fadeOutColor,
    fadeOutColor,3)
468                         if fadeOutCtr > 25:
469
470                             boxWidth = int(266.66 -
    2*10)
471                             boxHeight = 240-2*10
472                             fontSize = 35
473                             fadeOutCtr = 0
474
475                             DexHome.selectionMade=
    False
476
477                             if DexHome.
    selectedOption == 6: DexHome.selectedScreen = 1
478                             if DexHome.
    selectedOption == 7: pass
479                             if DexHome.
    selectedOption == 8: pass
480                             if DexHome.
    selectedOption == 9: pass
481                             if DexHome.
    selectedOption == 10: pass
482                             if DexHome.
    selectedOption == 11: pass
483
484                             repressCooldown = 10
485
486                         fadeOutCtr += 1
487
```

```
488                         if repressCooldown > 0:
    repressCooldown -= 1
489
490                     # Update Screen
491                     pygame.display.update()
492                     clock.tick(60)
493
494             DexHome.running = True
495             return
```

```
 1 # Importing Modules
 2 import pygame
 3 from pygame import gfxdraw
 4 from pygame.locals import *
 5 from threading import Thread
 6 import time
 7 import random
 8 import sys
 9 import sqlite3
10 import os
11
12 from CButton import Button
13 from SpriteManager import Sprite
14 from CDrawing import Draw
15 from CText import Text
16 from CUserInterface import UI
17
18
19
20
21 class DexInfo:
22
23 ##########################################################
   ######################################
24 #   VARIAVEIS

                            #
25 ##########################################################
   ######################################
26
27     pokeData = None
28     formData = None
29     megaData = None
30     megaDataSingle = None
31     formDataAll = None
32
33     evoChain = None
34     currentPokemon = 1
35     running = True
36     loadNewPokemon = False
37     evoScreenActive = False
38     evoSelectActive = False
39     statsScreenActive = False
40
```

```
41      formNumberSelected = None
42
43      megaEvolutionSelected = False
44      megaEvolutionNumber = 0
45
46      alolaFormSelected = False
47
48      galarianFormSelected = False
49
50      genderFemaleSelected = False
51
52      shinySelected = False
53
54
55      conn = sqlite3.connect('pokemon.db')
56      conn.row_factory = sqlite3.Row
57      c = conn.cursor()
58
59      thread = Thread()
60      sleepThread = Thread()
61
62  ##########################################################
    ######################################
63  #   TOGGLE FUNCTION

                    #
64  ##########################################################
    ######################################
65
66      def ReturnToMenu():
67          DexInfo.loadNewPokemon = True
68          DexInfo.running = False
69
70      def ToggleNextForm():
71          DexInfo.formNumberSelected += 1
72          if DexInfo.formNumberSelected > len(DexInfo.
    formDataAll): DexInfo.formNumberSelected = 1
73
74          DexInfo.loadNewPokemon = True
75
76          DexInfo.LoadSpritesheet()
77
78      def TogglePrevForm():
79          DexInfo.formNumberSelected -= 1
```

```
 80         if DexInfo.formNumberSelected < 1: DexInfo.
    formNumberSelected = len(DexInfo.formDataAll)
 81
 82         DexInfo.loadNewPokemon = True
 83
 84         DexInfo.LoadSpritesheet()
 85
 86     def ToggleAlolaForm():
 87         if DexInfo.alolaFormSelected: DexInfo.
    alolaFormSelected = False
 88         else: DexInfo.alolaFormSelected = True
 89         DexInfo.loadNewPokemon = True
 90
 91         DexInfo.LoadSpritesheet()
 92
 93     def ToggleGalarianForm():
 94         if DexInfo.galarianFormSelected: DexInfo.
    galarianFormSelected = False
 95         else: DexInfo.galarianFormSelected = True
 96         DexInfo.loadNewPokemon = True
 97
 98         DexInfo.LoadSpritesheet()
 99
100     def ToggleShinyOn():
101         DexInfo.shinySelected = True
102         DexInfo.oneTimeCycleLoad = True
103         DexInfo.LoadSpritesheet()
104
105     def ToggleShinyOff():
106         DexInfo.shinySelected = False
107         DexInfo.oneTimeCycleLoad = True
108         DexInfo.LoadSpritesheet()
109
110     def ToggleMegaEvolution1():
111         if DexInfo.megaEvolutionNumber == 1 and
    DexInfo.megaEvolutionSelected == True: DexInfo.
    megaEvolutionSelected = False
112         else: DexInfo.megaEvolutionSelected = True
113         DexInfo.megaEvolutionNumber = 1
114         DexInfo.loadNewPokemon = True
115
116         DexInfo.LoadSpritesheet()
117
118     def ToggleMegaEvolution2():
```

```
119         if DexInfo.megaEvolutionNumber == 2 and
    DexInfo.megaEvolutionSelected == True: DexInfo.
    megaEvolutionSelected = False
120         else: DexInfo.megaEvolutionSelected = True
121         DexInfo.megaEvolutionNumber = 2
122         DexInfo.loadNewPokemon = True
123 
124         DexInfo.LoadSpritesheet()
125 
126     def ToggleNormalMale():
127         DexInfo.shinySelected = False
128         DexInfo.genderFemaleSelected = False
129         DexInfo.megaEvolutionSelected = False
130         DexInfo.loadNewPokemon = True
131         DexInfo.oneTimeCycleLoad = True
132 
133         DexInfo.LoadSpritesheet()
134 
135     def ToggleNormalFemale():
136         DexInfo.shinySelected = False
137         DexInfo.genderFemaleSelected = True
138         DexInfo.megaEvolutionSelected = False
139         DexInfo.loadNewPokemon = True
140         DexInfo.oneTimeCycleLoad = True
141 
142         DexInfo.LoadSpritesheet()
143 
144     def ToggleShinyMale():
145         DexInfo.shinySelected = True
146         DexInfo.genderFemaleSelected = False
147         DexInfo.megaEvolutionSelected = False
148         DexInfo.loadNewPokemon = True
149         DexInfo.oneTimeCycleLoad = True
150 
151         DexInfo.LoadSpritesheet()
152 
153     def ToggleShinyFemale():
154         DexInfo.shinySelected = True
155         DexInfo.genderFemaleSelected = True
156         DexInfo.megaEvolutionSelected = False
157         DexInfo.loadNewPokemon = True
158         DexInfo.oneTimeCycleLoad = True
159 
160         DexInfo.LoadSpritesheet()
```

```
161
162     def ToggleStatsScreen():
163         if DexInfo.statsScreenActive: DexInfo.
    statsScreenActive = False
164         else: DexInfo.statsScreenActive = True
165         DexInfo.oneTimeCycleLoad = True
166
167         DexInfo.evoScreenActive = False
168         DexInfo.evoSelectActive = False
169
170     def ToggleEvoSelector():
171         if DexInfo.evoSelectActive: DexInfo.
    evoSelectActive = False
172         else: DexInfo.evoSelectActive = True
173         DexInfo.oneTimeCycleLoad = True
174
175         DexInfo.evoScreenActive = False
176         DexInfo.statsScreenActive = False
177
178     def ToggleEvoChainScreen():
179         if DexInfo.evoScreenActive: DexInfo.
    evoScreenActive = False
180         else: DexInfo.evoScreenActive = True
181         DexInfo.oneTimeCycleLoad = True
182
183         DexInfo.evoSelectActive = False
184         DexInfo.statsScreenActive = False
185
186     def ToggleNextDex():
187         DexInfo.formNumberSelected = None
188         DexInfo.alolaFormSelected = False
189         DexInfo.loadNewPokemon = True
190         DexInfo.evoSelectActive = False
191         DexInfo.megaEvolutionSelected = False
192         DexInfo.currentPokemon += 1
193         if DexInfo.currentPokemon >= 803: DexInfo.
    currentPokemon = 1
194
195         DexInfo.LoadSpritesheet()
196
197     def TogglePrevDex():
198         DexInfo.formNumberSelected = None
199         DexInfo.alolaFormSelected = False
200         DexInfo.loadNewPokemon = True
```

```
201         DexInfo.evoSelectActive = False
202         DexInfo.megaEvolutionSelected = False
203         DexInfo.currentPokemon -= 1
204         if DexInfo.currentPokemon <=  0: DexInfo.
    currentPokemon = 802
205 
206         DexInfo.LoadSpritesheet()
207 
208     def ToggleNextEvo():
209         DexInfo.formNumberSelected = None
210         DexInfo.alolaFormSelected = False
211         DexInfo.loadNewPokemon = True
212         DexInfo.evoSelectActive = False
213         DexInfo.megaEvolutionSelected = False
214         if DexInfo.pokeData["nextEvolution"] != None
    :
215             DexInfo.currentPokemon = DexInfo.
    pokeData["nextEvolution"]
216 
217         DexInfo.LoadSpritesheet()
218 
219     def TogglePrevEvo():
220         DexInfo.formNumberSelected = None
221         DexInfo.alolaFormSelected = False
222         DexInfo.loadNewPokemon = True
223         DexInfo.evoSelectActive = False
224         DexInfo.megaEvolutionSelected = False
225         if DexInfo.pokeData["prevEvolution"] != None
    :
226             DexInfo.currentPokemon = DexInfo.
    pokeData["prevEvolution"]
227 
228         DexInfo.LoadSpritesheet()
229 
230     def ToggleDexNumber(dexNumber):
231         DexInfo.formNumberSelected = None
232         DexInfo.alolaFormSelected = False
233         DexInfo.loadNewPokemon = True
234         DexInfo.evoSelectActive = False
235         DexInfo.megaEvolutionSelected = False
236         DexInfo.currentPokemon = dexNumber
237 
238         DexInfo.LoadSpritesheet()
239 
```

```
240     def GetPokeData(nationalDex):
241         parameters = (nationalDex,)
242         DexInfo.c.execute("""SELECT *,
243                 evoNext.evoNextDex AS nextEvolution,
244                 evoPrev.evoDex AS prevEvolution,
245                 typeA.typeName AS type1Name,
246                 typeB.typeName AS type2Name
247                 FROM pokemon
248                 LEFT JOIN sprites ON pokemon.
    nationalDex = sprites.nationalDex
249                 AND (sprites.isMegaEvolution IS NULL
     OR sprites.isMegaEvolution = '')
250                 LEFT JOIN types AS typeA ON pokemon.
    typeID1 = typeA.id
251                 LEFT JOIN types AS typeB ON pokemon.
    typeID2 = typeB.id
252                 LEFT JOIN regions ON pokemon.
    regionID = regions.id
253                 LEFT JOIN evYields ON pokemon.
    nationalDex = evYields.nationalDex
254                 LEFT JOIN evYieldTypes ON evYields.
    evYieldTypeID = evYieldTypes.id
255                 LEFT JOIN growthRates ON pokemon.
    growthRateID = growthRates.id
256                 LEFT JOIN eggGroups ON pokemon.
    eggGroupID = eggGroups.id
257                 LEFT JOIN evolutions AS evoNext ON
    pokemon.nationalDex = evoNext.evoDex
258                 LEFT JOIN evolutions AS evoPrev ON
    pokemon.nationalDex = evoPrev.evoNextDex
259                 WHERE pokemon.nationalDex = ?
260                 """,parameters)
261         return DexInfo.c.fetchone()
262 
263     def GetMegaData(nationalDex):
264         parameters = (nationalDex,)
265         DexInfo.c.execute("""SELECT *,
266                 typeA.typeName AS type1Name,
267                 typeB.typeName AS type2Name
268                 FROM pokemon
269                 LEFT JOIN pokemonMega ON pokemon.
    nationalDex = pokemonMega.nationalDex
270                 LEFT JOIN megaStones ON pokemonMega.
    megaStoneID = megaStones.id
```

```
271                 LEFT JOIN sprites ON pokemon.
    nationalDex = sprites.nationalDex
272                 AND sprites.isMegaEvolution = 1
273                 LEFT JOIN types AS typeA ON
    pokemonMega.megaTypeID1 = typeA.id
274                 LEFT JOIN types AS typeB ON
    pokemonMega.megaTypeID2 = typeB.id
275                 WHERE pokemon.nationalDex = ?
276                 ORDER BY megaName ASC
277                 """,parameters)
278         # Duplicate entries because of left join
279         result = DexInfo.c.fetchall()
280         if len(result) > 1: return [result[0],result
    [3]]
281         else: return result
282 
283     def GetAlolaData(nationalDex):
284         parameters = (nationalDex,)
285         DexInfo.c.execute("""SELECT *,
286                 typeA.typeName AS type1Name,
287                 typeB.typeName AS type2Name
288                 FROM pokemon
289                 LEFT JOIN sprites ON pokemon.
    nationalDex = sprites.nationalDex
290                 AND sprites.isAlolaForm = 1
291                 LEFT JOIN types AS typeA ON pokemon.
    typeID1 = typeA.id
292                 LEFT JOIN types AS typeB ON pokemon.
    typeID2 = typeB.id
293                 WHERE pokemon.nationalDex = ?
294                 """,parameters)
295         return DexInfo.c.fetchone()
296 
297     def GetGalarianData(nationalDex):
298         parameters = (nationalDex,)
299         DexInfo.c.execute("""SELECT *,
300                 typeA.typeName AS type1Name,
301                 typeB.typeName AS type2Name
302                 FROM pokemon
303                 LEFT JOIN sprites ON pokemon.
    nationalDex = sprites.nationalDex
304                 AND sprites.isGalarianForm = 1
305                 LEFT JOIN types AS typeA ON pokemon.
    typeID1 = typeA.id
```

```
306                 LEFT JOIN types AS typeB ON pokemon.
    typeID2 = typeB.id
307                 WHERE pokemon.nationalDex = ?
308                 """,parameters)
309         return DexInfo.c.fetchone()
310 
311 
312     def GetFormData(nationalDex,formNumber):
313         parameters = (formNumber,formNumber,
    nationalDex,)
314         DexInfo.c.execute("""SELECT *,
315                 typeA.typeName AS type1Name,
316                 typeB.typeName AS type2Name
317                 FROM pokemon
318                 LEFT JOIN sprites ON pokemon.
    nationalDex = sprites.nationalDex
319                 AND sprites.formNumber = ?
320                 LEFT JOIN types AS typeA ON pokemon.
    typeID1 = typeA.id
321                 LEFT JOIN types AS typeB ON pokemon.
    typeID2 = typeB.id
322                 LEFT JOIN pokemonForms ON pokemon.
    nationalDex = pokemonForms.nationalDex
323                 AND pokemonForms.formNumber = ?
324                 WHERE pokemon.nationalDex = ?
325                 """,parameters)
326         return DexInfo.c.fetchone()
327 
328     def GetFormDataAll(nationalDex):
329         parameters = (nationalDex,)
330         DexInfo.c.execute("""SELECT *
331                 FROM pokemonForms
332                 WHERE nationalDex = ?
333                 """,parameters)
334         return DexInfo.c.fetchall()
335 
336     def LoadSpritesheet():
337 
338         pokeTmp = DexInfo.GetPokeData(DexInfo.
    currentPokemon)
339         megaTmp = DexInfo.GetMegaData(DexInfo.
    currentPokemon)
340         alolaTmp = DexInfo.GetAlolaData(DexInfo.
    currentPokemon)
```

```
341
342         if pokeTmp["hasMultipleForms"] == 1 and
    DexInfo.formNumberSelected == None: DexInfo.
    formNumberSelected = pokeTmp["defaultForm"]
343         formTmp = DexInfo.GetFormData(DexInfo.
    currentPokemon,DexInfo.formNumberSelected)
344
345
346
347         if DexInfo.shinySelected:
348         # SHINY
349             # Spritesheets for Multiple Forms
350             if pokeTmp["hasMultipleForms"] == 1:
351                 spriteFile = formTmp["
    spriteSheetHDFrontShiny"]
352
353             # Spritesheets for Mega-Evolutions
354             elif pokeTmp["hasMegaEvolution"] == 1
    and DexInfo.megaEvolutionSelected:
355                 if len(megaTmp) > 1:
356                     if DexInfo.megaEvolutionNumber
     == 1: spriteFile = megaTmp[0]["
    spriteSheetHDFrontShiny"]
357                     else: spriteFile = megaTmp[1]["
    spriteSheetHDFrontShiny"]
358                 else: spriteFile = megaTmp[0]["
    spriteSheetHDFrontShiny"]
359
360             # Spritesheets for Alola-Form
361             elif pokeTmp["hasAlolaForm"] == 1 and
    DexInfo.alolaFormSelected:
362                 spriteFile = alolaTmp["
    spriteSheetHDFrontShiny"]
363
364             # Spritesheets for Gender-Difference
365             elif pokeTmp["genderDifference"] == 1:
366                 if DexInfo.genderFemaleSelected:
    spriteFile = pokeTmp["spriteSheetHDFrontFemaleShiny"
    ]
367                 else: spriteFile = pokeTmp["
    spriteSheetHDFrontShiny"]
368
369             # Default Spritesheet
370             else:
```

```
371                 spriteFile = pokeTmp["
    spriteSheetHDFrontShiny"]
372         else:
373         # NORMAL
374             # Spritesheets for Multiple Forms
375             if pokeTmp["hasMultipleForms"] == 1:
376                 spriteFile = formTmp["
    spriteSheetHDFront"]
377 
378             # Spritesheets for Mega-Evolutions
379             elif pokeTmp["hasMegaEvolution"] == 1
    and DexInfo.megaEvolutionSelected:
380                 if len(megaTmp) > 1:
381                     if DexInfo.megaEvolutionNumber
     == 1: spriteFile = megaTmp[0]["spriteSheetHDFront"]
382                     else: spriteFile = megaTmp[1]["
    spriteSheetHDFront"]
383                 else: spriteFile = megaTmp[0]["
    spriteSheetHDFront"]
384 
385             # Spritesheets for Alola-Form
386             elif pokeTmp["hasAlolaForm"] == 1 and
    DexInfo.alolaFormSelected:
387                 spriteFile = alolaTmp["
    spriteSheetHDFront"]
388 
389             # Spritesheets for Gender-Difference
390             elif pokeTmp["genderDifference"] == 1:
391                 if DexInfo.genderFemaleSelected:
    spriteFile = pokeTmp["spriteSheetHDFrontFemale"]
392                 else: spriteFile = pokeTmp["
    spriteSheetHDFront"]
393 
394             # Default Spritesheet
395             else:
396                 spriteFile = pokeTmp["
    spriteSheetHDFront"]
397 
398         DexInfo.thread = Thread(target = Sprite.
    Create, args = ("spriteSheets/" + str(spriteFile),
    DexInfo.currentPokemon,))
399         DexInfo.thread.start()
400 
401     def EvoChain():
```

```
402         evoChain = [DexInfo.currentPokemon]
403 
404         # get previous evolutions
405         currentSelector = DexInfo.currentPokemon
406         while True:
407             parameters = (currentSelector,)
408             DexInfo.c.execute("""SELECT * FROM
    evolutions WHERE evoNextDex = ?""",parameters)
409             evoResult = DexInfo.c.fetchone()
410             if evoResult != None:
411                 evoChain.insert(0,evoResult["evoDex"
    ])
412                 currentSelector = evoResult["evoDex"
    ]
413             else: break
414 
415         # get next evolutions
416         currentSelector = DexInfo.currentPokemon
417         while True:
418             parameters = (currentSelector,)
419             DexInfo.c.execute("""SELECT * FROM
    evolutions WHERE evoDex = ?""",parameters)
420             evoResult = DexInfo.c.fetchone()
421             if evoResult != None:
422                 evoChain.append(evoResult["
    evoNextDex"])
423                 currentSelector = evoResult["
    evoNextDex"]
424             else: break
425 
426         return evoChain
427 
428     def FullChain():
429         simpleChain = DexInfo.EvoChain()
430         fullChain = [[simpleChain[0]]]
431         fullChainSegment = []
432 
433         for sCh in simpleChain:
434             currentSelector = sCh
435             fullChainSegment = []
436 
437             parameters = (currentSelector,)
438             DexInfo.c.execute("""SELECT * FROM
    evolutions WHERE evoDex = ?""",parameters)
```

```
439             evoResults = DexInfo.c.fetchall()
440             for evoResult in evoResults:
441                 if evoResult != None:
442                     fullChainSegment.append(
    evoResult["evoNextDex"])
443             if len(fullChainSegment) != 0:
444                 fullChain.append(fullChainSegment)
445 
446         return fullChain
447 
448     def NextEvolutions():
449         nextEvos = []
450         parameters = (DexInfo.currentPokemon,)
451         DexInfo.c.execute("""SELECT * FROM
    evolutions WHERE evoDex = ?""",parameters)
452         evoResults = DexInfo.c.fetchall()
453         for evoResult in evoResults:
454             if evoResult != None:
455                 nextEvos.append(evoResult["
    evoNextDex"])
456         return nextEvos
457 
458     def HasMultipleEvos():
459 
460         parameters = (DexInfo.currentPokemon,)
461         DexInfo.c.execute("SELECT * FROM evolutions
    WHERE evoDex = ?",parameters)
462         evoResults = DexInfo.c.fetchall()
463         if len(evoResults) > 1: return True
464         else: return False
465 
466     def GetTypeColors(dexNumber):
467         parameters = (dexNumber,)
468         DexInfo.c.execute("SELECT * FROM pokemon
    LEFT JOIN types AS typeA ON pokemon.typeID1 = typeA.
    id WHERE pokemon.nationalDex = ?",parameters)
469         pmResult = DexInfo.c.fetchone()
470 
471         colorA = (int(pmResult["typeColor"].split(
    ',')[0]), int(pmResult["typeColor"].split(',')[1]),
    int(pmResult["typeColor"].split(',')[2]))
472         colorB = (int(pmResult["typeColorBright"].
    split(',')[0]), int(pmResult["typeColorBright"].
    split(',')[1]), int(pmResult["typeColorBright"].
```

```
472 split(',')[2]))
473
474         return (colorA,colorB)
475
476     def MegaEvolutionCount():
477         parameters = (DexInfo.currentPokemon,)
478         # CHANGE TO pokemonMega-Database
479         DexInfo.c.execute("SELECT * FROM sprites
    WHERE nationalDex = ? AND isMegaEvolution = 1",
    parameters)
480         pmResult = DexInfo.c.fetchall()
481
482         if pmResult != None: return len(pmResult)
483         else: return 0
484
485 ##################################################
    ######################################
486 ##################################################
    ######################################
487 #   MAIN START

                        #
488 ##################################################
    ######################################
489 ##################################################
    ######################################
490
491     def Show(selectedPokemon):
492
493 ##################################################
    ######################################
494 #   INITIALISATION AND SETUP

         #
495 ##################################################
    ######################################
496
497         # PyGame Initialisation
498         clock = pygame.time.Clock()
499
500         # Window and Surface Initialisation
501         displayWidth = 800
502         displayHeight = 480
503
```

```
504
505         idleCtr = 0
506
507         flags = FULLSCREEN | DOUBLEBUF
508
509         try:
510             if os.uname()[1] == 'raspberrypi':
511                 mainSurface = pygame.display.
    set_mode((0,0),flags)
512                 pygame.mouse.set_cursor((8,8),(0,0
    ),(0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,0))
513             else:
514                 mainSurface = pygame.display.
    set_mode((displayWidth,displayHeight))
515                 pygame.mouse.set_visible(True)
516         except:
517             mainSurface = pygame.display.set_mode((
    displayWidth,displayHeight))
518             pygame.mouse.set_visible(True)
519
520         pygame.event.set_allowed([QUIT, KEYDOWN,
    KEYUP])
521
522 ##################################################
    ######################################
523 #   SURFACE DEFINITIONS

                #
524 ##################################################
    ######################################
525
526         spriteSurface = pygame.Surface((300,300)).
    convert_alpha()
527
528         evoChainSurface = pygame.Surface((500-26,360
    -26)).convert_alpha()
529
530 ##################################################
    ######################################
531 #   VARIABLE DEFINITIONS

                #
532 ##################################################
    ######################################
```

```
533
534         runtimeCtr = 0
535
536         loadActiveCounter = 0
537         spriteReloadTrigger = 10
538         spriteReloaded = False
539
540         thread = None
541
542 ##################################################
    ####################################
543 #   LOADING LOOP

                        #
544 ##################################################
    ####################################
545
546         # Loading min and max values for stats
547         statMinMaxVals = []
548         statTypes = ("statHP","statAtk","statDef","
    statSpAtk","statSpDef","statSpd")
549         for stat in statTypes:
550             statSegment = []
551             DexInfo.c.execute("SELECT * FROM pokemon
     WHERE " + stat + " NOT NULL ORDER BY " + stat + "
    ASC")
552             statResMin = DexInfo.c.fetchone()
553             statSegment.append(statResMin[stat])
554             DexInfo.c.execute("SELECT * FROM pokemon
     ORDER BY " + stat + " DESC")
555             statResMax= DexInfo.c.fetchone()
556             statSegment.append(statResMax[stat])
557             statMinMaxVals.append(statSegment)
558
559         DexInfo.currentPokemon = selectedPokemon
560
561         DexInfo.LoadSpritesheet()
562
563         while DexInfo.running:
564
565             # Loading data
566             DexInfo.pokeData = DexInfo.GetPokeData(
    DexInfo.currentPokemon)
567
```

```
568             if DexInfo.pokeData["hasMultipleForms"
    ] == 1 and DexInfo.formNumberSelected == None:
    DexInfo.formNumberSelected = DexInfo.pokeData["
    defaultForm"]
569 
570             DexInfo.formData = DexInfo.GetFormData(
    DexInfo.currentPokemon,DexInfo.formNumberSelected)
571 
572             DexInfo.formDataAll = DexInfo.
    GetFormDataAll(DexInfo.currentPokemon)
573 
574             DexInfo.megaData = DexInfo.GetMegaData(
    DexInfo.currentPokemon)
575 
576             DexInfo.alolaData = DexInfo.GetAlolaData
    (DexInfo.currentPokemon)
577 
578             # Set default Form
579 
580             if DexInfo.megaEvolutionSelected:
581                 if len(DexInfo.megaData) > 1:
582                     if DexInfo.megaEvolutionNumber
     == 1: DexInfo.megaDataSingle = DexInfo.megaData[0]
583                     else: DexInfo.megaDataSingle =
    DexInfo.megaData[1]
584                 else: DexInfo.megaDataSingle =
    DexInfo.megaData[0]
585 
586             dexTypeColor = (int(DexInfo.pokeData["
    typeColor"].split(',')[0]), int(DexInfo.pokeData["
    typeColor"].split(',')[1]), int(DexInfo.pokeData["
    typeColor"].split(',')[2]))
587             dexTypeColorDark = (int(DexInfo.pokeData
    ["typeColorBright"].split(',')[0]), int(DexInfo.
    pokeData["typeColorBright"].split(',')[1]), int(
    DexInfo.pokeData["typeColorBright"].split(',')[2]))
588 
589             # Button Setup
590             Button.idleColor = dexTypeColor
591             Button.hoverColor = dexTypeColorDark
592             Button.fontColor = (255,255,255)
593             Button.disabledColor = (150,150,150)
594             Button.borderColor = (255,255,255)
595             Button.fontFamily = "joy.otf"
```

```
596
597             # Nav Buttons
598             btnPrevDex = Button.RoundRect(
    mainSurface,(320,425,110,40),15,"Anterior",18,1,
    DexInfo.TogglePrevDex,None,None,None,60,5)
599             btnPrevEvo = Button.RoundRect(
    mainSurface,(440,425,110,40),15,"Pre-Evo",18,1,
    DexInfo.TogglePrevEvo,None,None,None,60,5)
600             btnNextEvo = Button.RoundRect(
    mainSurface,(560,425,110,40),15,"Next Evo",18,1,
    DexInfo.ToggleNextEvo,None,None,None,60,5)
601             btnNextEvoSelect = Button.RoundRect(
    mainSurface,(560,425,110,40),15,"Next Evoo",18,1,
    DexInfo.ToggleEvoSelector,None,None,None,60,5)
602             btnReturn = Button.RoundRect(mainSurface
    ,(15,80,80,40),18,"< Voltar",16,1,DexInfo.
    ReturnToMenu,None,None,None,60,5)
603
604             btnNextDex = Button.RoundRect(
    mainSurface,(680,425,110,40),15,"Próximo",18,1,
    DexInfo.ToggleNextDex,None,None,None,60,5)
605
606             # Gender Buttons
607             btnFormNormal = Button.RoundRect(
    mainSurface,(520,150,126,40),15,"Normal",20,1,
    DexInfo.ToggleShinyOff,None,None,None,10)
608             btnFormShiny = Button.RoundRect(
    mainSurface,(661,150,126,40),15,"Shiny",20,1,DexInfo
    .ToggleShinyOn,None,None,None,10)
609
610             btnFormNormalMale = Button.RoundRect(
    mainSurface,(520,150,60,40),15,"M",25,1,DexInfo.
    ToggleNormalMale,None,None,None,10)
611             btnFormNormalFemale = Button.RoundRect(
    mainSurface,(520 + 66,150,60,40),15,"F",25,1,DexInfo
    .ToggleNormalFemale,None,None,None,10)
612             btnFormShinyMale = Button.RoundRect(
    mainSurface,(661,150,60,40),15,"SM",25,1,DexInfo.
    ToggleShinyMale,None,None,None,10)
613             btnFormShinyFemale = Button.RoundRect(
    mainSurface,(661 + 66,150,60,40),15,"SF",25,1,
    DexInfo.ToggleShinyFemale,None,None,None,10)
614
615             # Alola
```

```
616             btnAlolaToggle = Button.RoundRect(
    mainSurface,(16,325,70,40),18,"Alola",18,1,DexInfo.
    ToggleAlolaForm,None,None,None,40,30)
617             # Galarian
618             """btnGalarianToggle = Button.RoundRect(
    mainSurface,(16,275,70,40),18,"Galarian",18,1,
    DexInfo.ToggleGalariamForm,Nome,Nome,Nome,40,30)"""
619
620             # Form Selectors
621             btnNextForm = Button.RoundRect(
    mainSurface,(465,160,40,80),18,">",18,1,DexInfo.
    ToggleNextForm,None,None,None,40,50)
622             btnPrevForm = Button.RoundRect(
    mainSurface,(15,160,40,80),18,"<",18,1,DexInfo.
    TogglePrevForm,None,None,None,40,50)
623
624             # MegaEvolution Buttons
625             btnMegaEvo1 = Button.RoundRect(
    mainSurface,(18,280,40,40),10,"ME1",25,1,DexInfo.
    ToggleMegaEvolution1,None,None,None,0,30)
626             btnMegaEvo2 = Button.RoundRect(
    mainSurface,(18,340,40,40),10,"ME2",25,1,DexInfo.
    ToggleMegaEvolution2,None,None,None,0,30)
627
628             # ScreenToggle Buttons
629             btnEvoChainScreen = Button.RoundRect(
    mainSurface,(520,200,270,25),10,"Cadeia Evolutiva",
    20,2,DexInfo.ToggleEvoChainScreen,None,None,None,10,
    10)
630             btnStatsScreen = Button.RoundRect(
    mainSurface,(525+115,90,146,45),15,"Mais Stats",17,
    None,DexInfo.ToggleStatsScreen,None,None,None,10)
631
632
633
634             # One-Time Drawing routines
635
636             mainSurface.fill((30,30,30))
637
638
639             Draw.RoundRect(mainSurface,(40,40,40),(
    520,10,270,130),15,2,dexTypeColor,"Stats")
640             Draw.RoundRect(mainSurface,(40,40,40),(
    525,40,110,45),10,1,dexTypeColor,"")
```

```
641             Text.Write(mainSurface,(525+55,50),"
    Tamanho",17,"joy.otf",(255,255,255),True)
642             Text.Write(mainSurface,(525+55,73),str(
    '{0:.2f}'.format(DexInfo.pokeData["height"]/10)) +
    " m",19,"joy.otf",(255,255,255),True)
643             Draw.RoundRect(mainSurface,(40,40,40),(
    525,90,110,45),10,1,dexTypeColor,"")
644             Text.Write(mainSurface,(525+55,100),"
    Peso",17,"joy.otf",(255,255,255),True)
645             Text.Write(mainSurface,(525+55,123),str(
    '{0:.2f}'.format(DexInfo.pokeData["weight"]/10)) +
    " kg",19,"joy.otf",(255,255,255),True)
646             Draw.RoundRect(mainSurface,(40,40,40),(
    525+115,40,146,45),10,1,dexTypeColor,"")
647             Text.Write(mainSurface,(525+115+73,50),"
    Egg Group",17,"joy.otf",(255,255,255),True)
648             Text.Write(mainSurface,(525+115+73,73),
    DexInfo.pokeData["eggGroupName"],17,"joy.otf",(255,
    255,255),True)
649             Draw.RoundRect(mainSurface,(40,40,40),(
    520,200,270,100),15,2,dexTypeColor,"Cadeia Evolutiva
    ")
650             Draw.RoundRect(mainSurface,(40,40,40),(
    520,310,270,60),15,2,dexTypeColor,"Gender Ratio")
651             Draw.RoundRect(mainSurface,(40,40,40),(
    10,360,300,110),15,2,dexTypeColor)
652             Draw.RoundRect(mainSurface,dexTypeColor
    ,(10,300,400,115),15,2,dexTypeColor)
653             Draw.RoundRect(mainSurface,(40,40,40),(
    10,10,500,360),26,2,dexTypeColor)
654
655
656             # Evo Chain
657             evoImg = pygame.image.load("sprites/" +
    str('{0:03d}'.format(DexInfo.pokeData["nationalDex"
    ])) +"/sprite-small-FN-" + str('{0:03d}'.format(
    DexInfo.pokeData["nationalDex"])) + ".png")
658             evoImg = pygame.transform.scale(evoImg,(
    96,96))
659             mainSurface.blit(evoImg,(610,218))
660
661             # Gender Ratio
662             totalBarWidth = 255
663             widthMale = DexInfo.pokeData["genderMale
```

```
663 "]*(totalBarWidth/100)
664             widthFemale = (100-DexInfo.pokeData["
    genderFemale"])*(totalBarWidth/100)
665 
666             if DexInfo.pokeData["genderMale"] < 100
    and DexInfo.pokeData["genderMale"] > 0:
667                 pygame.draw.rect(mainSurface,
    dexTypeColor,(5+widthMale+520,335,5,35))
668                 Text.Write(mainSurface,(5+520+(
    widthMale/2),352),"M",25,"joy.otf",(255,255,255),
    True)
669                 Text.Write(mainSurface,(5+5+5+
    widthMale+520+(widthFemale/2),352),"F",25,"joy.otf"
    ,(255,255,255),True)
670             elif DexInfo.pokeData["genderMale"] >=
    100:
671                 Text.Write(mainSurface,(520+5+(255/2
    ),352),"M",25,"joy.otf",(255,255,255),True)
672             else:
673                 Text.Write(mainSurface,(520+5+(255/2
    ),352),"F",25,"joy.otf",(255,255,255),True)
674 
675             Draw.RoundRect(mainSurface,(40,40,40),(
    20,374,110,39),10)
676             Draw.Pokeball(mainSurface,(35,35),
    dexTypeColor,(40,40,40))
677 
678             Text.Write(mainSurface,(28,376),"#" +
    str('{0:03d}'.format(DexInfo.pokeData["nationalDex"
    ])),35,"joy.otf",(255,255,255))
679 
680             if DexInfo.megaEvolutionSelected: Text.
    Write(mainSurface,(138,382),DexInfo.megaDataSingle["
    megaName"],25,"joy.otf",(255,255,255))
681             else: Text.Write(mainSurface,(138,376),
    DexInfo.pokeData["name"],35,"joy.otf",(255,255,255))
682 
683             Text.Write(mainSurface,(20,425),"Species
    :",20,"calibrilight.ttf",(255,255,255))
684             Text.Write(mainSurface,(20,445),"Region
    :",20,"calibrilight.ttf",(255,255,255))
685             Text.Write(mainSurface,(90,425),DexInfo.
    pokeData["species"],20,"calibrilight.ttf",(255,255,
    255))
```

```
686             Text.Write(mainSurface,(90,445),DexInfo.
    pokeData["regionName"],20,"calibrilight.ttf",(255,
    255,255))
687             Text.Write(mainSurface,(180,445),"
    Generation:",20,"calibrilight.ttf",(255,255,255))
688             Text.Write(mainSurface,(280,445),str(
    DexInfo.pokeData["regionID"]),20,"calibrilight.ttf"
    ,(255,255,255))
689 
690             pygame.draw.rect(mainSurface,
    dexTypeColor,(420,383,12,12))
691             pygame.draw.rect(mainSurface,
    dexTypeColor,(436,383,24,12))
692             pygame.draw.rect(mainSurface,
    dexTypeColor,(464,383,45,12))
693             Text.Write(mainSurface,(425,396),"T  Y
    P  E  :",18,"joy.otf",dexTypeColor)
694 
695             if DexInfo.megaEvolutionSelected:
696                 if DexInfo.megaDataSingle["type2Name
    "] == None or DexInfo.megaDataSingle["type2Name"
    ] == "":
697                     Draw.TypeSignSingle(mainSurface
    ,(520,380),dexTypeColor,DexInfo.megaDataSingle["
    type1Name"])
698                 else:
699                     Draw.TypeSign1(mainSurface,(520,
    380),dexTypeColor,DexInfo.megaDataSingle["type1Name"
    ])
700                     Draw.TypeSign2(mainSurface,(645,
    380),dexTypeColorDark,DexInfo.megaDataSingle["
    type2Name"])
701             else:
702                 if DexInfo.pokeData["type2Name"] ==
    None or DexInfo.pokeData["type2Name"] == "":
703                     Draw.TypeSignSingle(mainSurface
    ,(520,380),dexTypeColor,DexInfo.pokeData["type1Name"
    ])
704                 else:
705                     Draw.TypeSign1(mainSurface,(520,
    380),dexTypeColor,DexInfo.pokeData["type1Name"])
706                     Draw.TypeSign2(mainSurface,(645,
    380),dexTypeColorDark,DexInfo.pokeData["type2Name"])
707 
```

```
708
709
710
711             # Drawing Buttons before cycle (fixes
    visual bug)
712             # Nav Buttons
713             pygame.display.update(btnPrevDex.Show(
    False))
714             pygame.display.update(btnPrevEvo.Show(
    False))
715             pygame.display.update(btnNextEvo.Show(
    False))
716             pygame.display.update(btnNextDex.Show(
    False))
717
718             if not DexInfo.statsScreenActive and not
     DexInfo.evoSelectActive:
719                 pygame.display.update(btnReturn.Show
    (False))
720
721
722
723             # Gender & Form Buttons
724             if DexInfo.pokeData["genderDifference"
    ] == 1 and not DexInfo.alolaFormSelected:
725                 pygame.display.update(
    btnFormNormalMale.Show(False))
726                 pygame.display.update(
    btnFormNormalFemale.Show(False))
727                 pygame.display.update(
    btnFormShinyMale.Show(False))
728                 pygame.display.update(
    btnFormShinyFemale.Show(False))
729             else:
730                 pygame.display.update(btnFormNormal.
    Show(False))
731                 pygame.display.update(btnFormShiny.
    Show(False))
732
733             # Screen Select Buttons
734             pygame.display.update(btnEvoChainScreen.
    Show(False))
735             pygame.display.update(btnStatsScreen.
    Show(False))
```

```
736
737
738
739             DexInfo.evoChain = DexInfo.EvoChain()
740             DexInfo.fullChain = DexInfo.FullChain()
741
742             pygame.display.update()
743
744             spriteReloaded = False
745             DexInfo.loadNewPokemon = False
746
747             DexInfo.oneTimeCycleLoad = True
748
749 ##################################################
    ######################################
750 #   RUNNING LOOP

                        #
751 ##################################################
    ######################################
752
753             while not DexInfo.loadNewPokemon:
754
755                 # Event Processing
756                 for event in pygame.event.get():
757                     if event.type == pygame.QUIT:
758                         pygame.quit()
759                         sys.exit()
760
761                 # Keypress Processing
762                 keys = pygame.key.get_pressed()
763                 if keys[pygame.K_q] != 0:
764                     pygame.quit()
765                     sys.exit()
766
767                 mouse = pygame.mouse.get_pos()
768                 click = pygame.mouse.get_pressed()
769
770
771                 if DexInfo.pokeData["nextEvolution"
    ] != None: nextEvoExists = True
772                 else: nextEvoExists = False
773
774                 if DexInfo.pokeData["prevEvolution"
```

```
774 ] != None: prevEvoExists = True
775                 else: prevEvoExists = False
776 
777 ############### RENDER ACTIVE BUTTONS
     ##################################################
    ###############
778                 if not DexInfo.thread.isAlive():
779                     # Nav Buttons
780                     pygame.display.update(btnPrevDex
    .Show())
781                     pygame.display.update(btnPrevEvo
    .Show(disabled = not prevEvoExists))
782                     if DexInfo.HasMultipleEvos():
    pygame.display.update(btnNextEvoSelect.Show(disabled
     = not nextEvoExists))
783                     else: pygame.display.update(
    btnNextEvo.Show(disabled = not nextEvoExists))
784                     pygame.display.update(btnNextDex
    .Show())
785 
786                     if not DexInfo.statsScreenActive
     and not DexInfo.evoSelectActive:
787                         pygame.display.update(
    btnReturn.Show())
788 
789 
790                     # Gender & Form Buttons
791                     if DexInfo.pokeData["
    genderDifference"] == 1 and not DexInfo.
    alolaFormSelected:
792                         pygame.display.update(
    btnFormNormalMale.Show())
793                         pygame.display.update(
    btnFormNormalFemale.Show())
794                         pygame.display.update(
    btnFormShinyMale.Show())
795                         pygame.display.update(
    btnFormShinyFemale.Show())
796                     else:
797                         pygame.display.update(
    btnFormNormal.Show())
798                         pygame.display.update(
    btnFormShiny.Show())
799 
```

```
800                     # Screen Select Buttons
801                     pygame.display.update(
    btnEvoChainScreen.Show())
802                     pygame.display.update(
    btnStatsScreen.Show())
803 
804 ############### RENDER DUMMY BUTTONS
     ##################################################
    ################
805                 else:
806                     # Nav Buttons
807                     pygame.display.update(btnPrevDex
    .Show(False))
808                     pygame.display.update(btnPrevEvo
    .Show(False,not prevEvoExists))
809                     pygame.display.update(btnNextEvo
    .Show(False,not nextEvoExists))
810                     pygame.display.update(btnNextDex
    .Show(False))
811 
812                     if not DexInfo.statsScreenActive
     and not DexInfo.evoSelectActive:
813                         pygame.display.update(
    btnReturn.Show(False))
814 
815                     # Gender & Form Buttons
816                     if DexInfo.pokeData["
    genderDifference"] == 1 and not DexInfo.
    alolaFormSelected:
817                         pygame.display.update(
    btnFormNormalMale.Show(False))
818                         pygame.display.update(
    btnFormNormalFemale.Show(False))
819                         pygame.display.update(
    btnFormShinyMale.Show(False))
820                         pygame.display.update(
    btnFormShinyFemale.Show(False))
821                     else:
822                         pygame.display.update(
    btnFormNormal.Show(False))
823                         pygame.display.update(
    btnFormShiny.Show(False))
824 
825                     # Screen Select Buttons
```

```
826                         pygame.display.update(
    btnEvoChainScreen.Show(False))
827                         pygame.display.update(
    btnStatsScreen.Show(False))
828 
829                     # Animation-Cycle for the Sprite and
     Stat-Screens
830 ################ STATUS SCREENS
     ##################################################
    ######################
831                     if DexInfo.statsScreenActive:
832                         # Only load once
833                         if DexInfo.oneTimeCycleLoad:
834                             Draw.RoundRect(mainSurface,(
    40,40,40),(10,10,500,360),26,2,dexTypeColor)
835                             evoChainSurface.fill((40,40,
    40))
836                             evoChainSurface.set_colorkey
    ((0,0,0))
837                             Text.Write(evoChainSurface,(
    70,0),"Pokemon",25,"joy.otf",(255,255,255))
838                             Text.Write(evoChainSurface,(
    150,20),"stats",25,"joy.otf",(255,255,255))
839 
840                             Draw.RoundRect(
    evoChainSurface,(40,40,40),(5,50,250,282),15,2,
    dexTypeColor,"Base Stats")
841                             UI.ProgressBar(
    evoChainSurface,(25,110),180,15,dexTypeColor,"HP",
    statMinMaxVals[0][0],statMinMaxVals[0][1],DexInfo.
    pokeData["statHP"])
842                             UI.ProgressBar(
    evoChainSurface,(25,150),180,15,dexTypeColorDark,"
    Attack",statMinMaxVals[1][0],statMinMaxVals[1][1],
    DexInfo.pokeData["statAtk"])
843                             UI.ProgressBar(
    evoChainSurface,(25,190),180,15,dexTypeColor,"
    Defense",statMinMaxVals[2][0],statMinMaxVals[2][1],
    DexInfo.pokeData["statDef"])
844                             UI.ProgressBar(
    evoChainSurface,(25,230),180,15,dexTypeColorDark,"
    Special Attack",statMinMaxVals[3][0],statMinMaxVals[
    3][1],DexInfo.pokeData["statSpAtk"])
845                             UI.ProgressBar(
```

```
845 evoChainSurface,(25,270),180,15,dexTypeColor,"
    Special Defense",statMinMaxVals[4][0],statMinMaxVals
    [4][1],DexInfo.pokeData["statSpDef"])
846                             UI.ProgressBar(
    evoChainSurface,(25,310),180,15,dexTypeColorDark,"
    Speed",statMinMaxVals[5][0],statMinMaxVals[5][1],
    DexInfo.pokeData["statSpd"])
847
848                             Draw.RoundRect(
    evoChainSurface,(40,40,40),(265,2,205,150),15,2,
    dexTypeColor,"Training")
849
850                             Text.Write(evoChainSurface,(
    275,30),"Catch Rate:",15,"joy.otf",(180,180,180))
851                             Text.Write(evoChainSurface,(
    380,30),str(DexInfo.pokeData["catchRate"]) + "%",15,
    "joy.otf",(255,255,255))
852                             Text.Write(evoChainSurface,(
    275,50),"Base Friendship:",15,"joy.otf",(180,180,180
    ))
853                             Text.Write(evoChainSurface,(
    420,50),str(DexInfo.pokeData["baseFriendship"]),15,"
    joy.otf",(255,255,255))
854                             Text.Write(evoChainSurface,(
    275,70),"Base Exp.:",15,"joy.otf",(180,180,180))
855                             Text.Write(evoChainSurface,(
    365,70),str(DexInfo.pokeData["baseExp"]),15,"joy.otf
    ",(255,255,255))
856                             Text.Write(evoChainSurface,(
    275,90),"Growth.:",15,"joy.otf",(180,180,180))
857                             Text.Write(evoChainSurface,(
    355,90),str(DexInfo.pokeData["growthRate"]),15,"joy.
    otf",(255,255,255))
858
859                             Text.Write(evoChainSurface,(
    275,110),"EV-Yield:",15,"joy.otf",(180,180,180))
860                             # Fetch EV-Yield data
861                             parameters = (DexInfo.
    currentPokemon,)
862                             DexInfo.c.execute("SELECT *
    FROM evYields LEFT JOIN evYieldTypes ON evYields.
    evYieldTypeID = evYieldTypes.id WHERE evYields.
    nationalDex = ?",parameters)
863                             pmResult = DexInfo.c.
```

```
863 fetchall()
864                             evYieldTextOffset = 0
865                             for evYield in pmResult:
866                                 Text.Write(
    evoChainSurface,(360,110 + evYieldTextOffset),str(
    evYield["evYieldPoints"]) + " " + evYield["
    evYieldType"],15,"joy.otf",(255,255,255))
867                                 evYieldTextOffset += 20
868 
869 
870                             Draw.RoundRect(
    evoChainSurface,(40,40,40),(265,162,205,170),15,2,
    dexTypeColor,"Pokedex Entry")
871 
872                             Text.WriteMultiline(
    evoChainSurface,str(DexInfo.pokeData["dexInfo"]),(
    275,190),pygame.font.Font("calibrilight.ttf",18),(
    255,255,255))
873 
874                             mainSurface.blit(
    evoChainSurface,(23,23))
875                             Draw.Pokeball(mainSurface,(
    35,35),dexTypeColor,(40,40,40))
876                         pygame.display.update(0,0,526,
    366)
877 
878 
879 ############### EVO SELECTION SCREEN
     ##################################################
    ################
880                     elif DexInfo.evoSelectActive:
881                         # Only load once
882                         if DexInfo.oneTimeCycleLoad:
883                             Draw.RoundRect(mainSurface,(
    40,40,40),(10,10,500,360),26,2,dexTypeColor)
884 
885                             evoChainSurface.fill((40,40,
    40))
886                             evoChainSurface.set_colorkey
    ((0,0,0))
887                             if DexInfo.currentPokemon
     == 133: Text.Write(evoChainSurface,(250,30),"Select
     next eeveelution",25,"joy.otf",(255,255,255),True)
888                             else: Text.Write(
```

```
888 evoChainSurface,(250,30),"Select next evolution",25,
    "joy.otf",(255,255,255),True)
889                             mainSurface.blit(
    evoChainSurface,(23,23))
890 
891                         horizontalOffset = 10
892                         verticalOffset = 120
893 
894                         nextEvos = DexInfo.
    NextEvolutions()
895 
896                         if len(nextEvos) == 2:
    horizontalOffset = 140
897                         elif len(nextEvos) == 3:
    horizontalOffset = 80
898                         else: horizontalOffset = 10
899 
900                         if len(nextEvos) > 4:
    verticalOffset = 70
901 
902                         evoCount = 0
903 
904                         for evo in nextEvos:
905 
906                             typeColors = DexInfo.
    GetTypeColors(evo)
907 
908                             if horizontalOffset+23-2 <
    mouse[0] < horizontalOffset+110+23+2 and
    verticalOffset+23-2 < mouse[1] < verticalOffset+96+
    23+2:
909                                 Draw.RoundRect(
    evoChainSurface,(40,40,40),(horizontalOffset,
    verticalOffset,96,110),15,2,typeColors[1],"#" + str(
    '{0:03d}'.format(evo)))
910                                 if click[0] == 1:
    DexInfo.ToggleDexNumber(evo)
911 
912                             else:
913                                 Draw.RoundRect(
    evoChainSurface,(40,40,40),(horizontalOffset,
    verticalOffset,96,110),15,2,typeColors[0],"#" + str(
    '{0:03d}'.format(evo)))
914 
```

```
915                             nextEvoImg = pygame.
    transform.scale(pygame.image.load("sprites/" + str(
    '{0:03d}'.format(evo)) +"/sprite-small-FN-" + str('{
    0:03d}'.format(evo)) + ".png"),(96,96))
916                             evoChainSurface.blit(
    nextEvoImg,(horizontalOffset,verticalOffset+20))
917                             horizontalOffset += 120
918                             evoCount += 1
919                             if evoCount >= 4:
920                                 horizontalOffset = 10
921                                 verticalOffset += 120
922                                 evoCount = 0
923 
924 
925                         mainSurface.blit(evoChainSurface
    ,(23,23))
926                         Draw.Pokeball(mainSurface,(35,35
    ),dexTypeColor,(40,40,40))
927                         pygame.display.update(0,0,526,
    366)
928 
929 ############### EVO CHAIN SCREEN
     ##################################################
    ####################
930                     elif DexInfo.evoScreenActive:
931 
932                         # Only load once
933                         if DexInfo.oneTimeCycleLoad:
934                             Draw.RoundRect(mainSurface,(
    40,40,40),(10,10,500,360),26,2,dexTypeColor)
935 
936                             evoChainSurface.fill((40,40,
    40))
937                             evoChainSurface.set_colorkey
    ((0,0,0))
938                             Text.Write(evoChainSurface,(
    250,30),"Evolution Chain",25,"joy.otf",(255,255,255
    ),True)
939                             Text.Write(evoChainSurface,(
    250,320),str(len(DexInfo.fullChain)) + " stage
    evolution-chain",20,"joy.otf",(255,255,255),True)
940 
941                             verticalOffset = 0
942                             scaleFactor = 2
```

```
943                             scaleOffsetCorrection = 48
944                             scaleHorizontalCorrection =
    48
945
946                             if len(DexInfo.fullChain
    ) == 1: horizontalOffset = 205
947                             elif len(DexInfo.fullChain
    ) == 2: horizontalOffset = 205 - 48 - 40
948                             elif len(DexInfo.fullChain
    ) == 3: horizontalOffset = 205 - 96 - 60
949                             elif len(DexInfo.fullChain
    ) == 4: horizontalOffset = 205 - 144 - 75
950
951                             if len(DexInfo.fullChain) >
    2:
952                                 scaleFactor = 1
953                                 scaleOffsetCorrection =
    0
954
    scaleHorizontalCorrection = 0
955
956                             for evoGroups in DexInfo.
    fullChain:
957                                 if len(evoGroups) > 1:
958                                     scaleFactor = 1
959
    scaleOffsetCorrection = 0
960
    scaleHorizontalCorrection = 0
961
962                             if len(DexInfo.fullChain
    ) == 2 and scaleFactor == 2: horizontalOffset -= 20
963
964                             for evoGroups in DexInfo.
    fullChain:
965
966                                 if len(evoGroups) == 1:
    verticalOffset = 131
967                                 elif len(evoGroups) == 2
    :verticalOffset = 83
968                                 else: verticalOffset =
    35
969
970                                 evoItemCount = 0
```

```
971                                 for evoItem in
    evoGroups:
972                                     evoItemImg = pygame
    .transform.scale(pygame.image.load("sprites/" + str
    ('{0:03d}'.format(evoItem)) +"/sprite-small-FN-" +
    str('{0:03d}'.format(evoItem)) + ".png"),(96*
    scaleFactor,96*scaleFactor))
973                                     evoChainSurface.
    blit(evoItemImg,(horizontalOffset-
    scaleOffsetCorrection,verticalOffset-
    scaleOffsetCorrection))
974                                     verticalOffset +=
    96
975                                     evoItemCount += 1
976                                     if evoItemCount >=
    3:
977
    horizontalOffset += 60
978                                         verticalOffset
     = 35
979                                         evoItemCount =
    0
980                                 horizontalOffset += 150
     + scaleHorizontalCorrection
981
982                             mainSurface.blit(
    evoChainSurface,(23,23))
983
984                             Draw.Pokeball(mainSurface,(
    35,35),dexTypeColor,(40,40,40))
985                             pygame.display.update(0,0,
    526,366)
986
987 ################ ANIMATED SPRITE CYCLE
     ##############################################
    ################
988                 else:
989
990                     # Form Buttons
991                     if DexInfo.pokeData["
    hasMultipleForms"] == 1:
992
993                         pygame.display.update(
    btnNextForm.Show())
```

```
 994                         pygame.display.update(
     btnPrevForm.Show())
 995
 996                     # Alola Form Button
 997                     if DexInfo.pokeData["
     hasAlolaForm"] == 1:
 998                         pygame.display.update(
     btnAlolaToggle.Show())
 999
1000                     # Mega Evolution Buttons
1001                     if DexInfo.pokeData["
     hasMegaEvolution"] != None:
1002                         megaStoneImg = pygame.
     transform.scale(pygame.image.load("megaStones/" +
     DexInfo.megaData[0]["megaStoneImage"]),(80,80))
1003                         mainSurface.blit(
     megaStoneImg,(10,290))
1004                         if 10 < mouse[0] < 90 and
     290 < mouse[1] < 370:
1005                             Text.Write(mainSurface
     ,(50,330),"Mega",18,"joy.otf",dexTypeColor,True)
1006                             if not DexInfo.
     sleepThread.isAlive() and click[0] == 1:
1007                                 DexInfo.
     ToggleMegaEvolution1()
1008                                 DexInfo.sleepThread
      = Thread(target = time.sleep, args = (0.3,))
1009                                 DexInfo.sleepThread
     .start()
1010                         else: Text.Write(
     mainSurface,(50,330),"Mega",18,"joy.otf",(0,0,0),
     True)
1011                         pygame.display.update((14,
     274,92,92))
1012
1013                         if len(DexInfo.megaData) >
     1:
1014                             megaStoneImg = pygame.
     transform.scale(pygame.image.load("megaStones/" +
     DexInfo.megaData[1]["megaStoneImage"]),(80,80))
1015                             mainSurface.blit(
     megaStoneImg,(10,220))
1016                             if 10 < mouse[0] < 90
     and 220 < mouse[1] < 300:
```

```
1017                                    Text.Write(
    mainSurface,(50,260),"Mega",18,"joy.otf",
    dexTypeColor,True)
1018                                    if not DexInfo.
    sleepThread.isAlive() and click[0] == 1:
1019                                        DexInfo.
    ToggleMegaEvolution2()
1020                                        DexInfo.
    sleepThread = Thread(target = time.sleep, args = (0
    .3,))
1021                                        DexInfo.
    sleepThread.start()
1022                                    else: Text.Write(
    mainSurface,(50,260),"Mega",18,"joy.otf",(0,0,0),
    True)
1023                                    pygame.display.update((
    14,204,92,92))
1024
1025
1026
1027                        if DexInfo.oneTimeCycleLoad:
1028                            # Sprite-Box
1029                            Draw.RoundRect(mainSurface
    ,(40,40,40),(10,10,500,360),26,2,dexTypeColor)
1030                            Draw.Pokeball(mainSurface,(
    35,35),dexTypeColor,(20,20,20))
1031                            Draw.RoundRect(mainSurface
    ,(20,20,20),(412,18,90,90),21,2,dexTypeColor)
1032                            spriteImg = pygame.image.
    load("sprites/" + str('{0:03d}'.format(DexInfo.
    pokeData["nationalDex"])) +"/sprite-small-FN-" +
    str('{0:03d}'.format(DexInfo.pokeData["nationalDex"
    ])) + ".png")
1033                            spriteImg = pygame.
    transform.scale(spriteImg,(96,96))
1034                            mainSurface.blit(spriteImg
    ,(412-3,18-3))
1035                            Text.Write(mainSurface,(456
    ,118),"Show Sprites",12,"joy.otf",(255,255,255),
    True)
1036
1037                            if DexInfo.pokeData["
    hasMultipleForms"] == 1: Text.Write(mainSurface,(
    250,350),"Form: " + DexInfo.formData["formName"],25
```

```
1037 ,"joy.otf",(255,255,255),True)
1038                             if DexInfo.shinySelected:
     Text.Write(mainSurface,(470,350),"S",30,"joy.otf",(
     255,255,255),True)
1039                             pygame.display.update(0,0,
     526,386)
1040 
1041                         if not DexInfo.thread.isAlive
     () and Sprite.loadedSpriteNr == DexInfo.
     currentPokemon:
1042 
1043                             if runtimeCtr % 1 == 0:
1044                                 Sprite.Cycle(Sprite.
     frameIndex,Sprite.tilesAmount,Sprite.frames)
1045                                 spriteSurface.fill((40,
     40,40))
1046                                 spriteSurface.
     set_colorkey((0,0,0))
1047                                 spriteSurface.blit(
     Sprite.current,Sprite.sprite)
1048                                 mainSurface.blit(
     spriteSurface,(100,30))
1049                                 pygame.display.update(
     100,30,300,300)
1050                         else:
1051                             spriteSurface.fill((40,40,
     40))
1052                             spriteSurface.set_colorkey
     ((0,0,0))
1053                             if runtimeCtr % 2 == 0:
1054                                 pygame.gfxdraw.aacircle
     (spriteSurface,120,150,8,dexTypeColor)
1055                                 pygame.gfxdraw.aacircle
     (spriteSurface,150,150,8,dexTypeColorDark)
1056                                 pygame.gfxdraw.aacircle
     (spriteSurface,180,150,8,dexTypeColor)
1057                                 pygame.gfxdraw.
     filled_circle(spriteSurface,120,150,8,dexTypeColor)
1058                                 pygame.gfxdraw.
     filled_circle(spriteSurface,150,150,8,
     dexTypeColorDark)
1059                                 pygame.gfxdraw.
     filled_circle(spriteSurface,180,150,8,dexTypeColor)
1060                             else:
```

```
1061                                pygame.gfxdraw.aacircle
     (spriteSurface,120,150,8,dexTypeColorDark)
1062                                pygame.gfxdraw.aacircle
     (spriteSurface,150,150,8,dexTypeColor)
1063                                pygame.gfxdraw.aacircle
     (spriteSurface,180,150,8,dexTypeColorDark)
1064                                pygame.gfxdraw.
     filled_circle(spriteSurface,120,150,8,
     dexTypeColorDark)
1065                                pygame.gfxdraw.
     filled_circle(spriteSurface,150,150,8,dexTypeColor)
1066                                pygame.gfxdraw.
     filled_circle(spriteSurface,180,150,8,
     dexTypeColorDark)
1067                            pygame.draw.rect(
     mainSurface,(40,40,40),(22,330,476,36))
1068                            mainSurface.blit(
     spriteSurface,(100,30))
1069
1070
1071                            pygame.display.update(100,
     30,300,300)
1072
1073                    clock.tick(60)
1074
1075                    runtimeCtr += 1
1076                    if runtimeCtr > 10000: runtimeCtr
      = 1
1077
1078                    DexInfo.oneTimeCycleLoad = False
1079
1080                    # Re-Triggers sprite-loading if
     thread failed
1081                    if Sprite.loadedSpriteNr != DexInfo
     .currentPokemon: loadActiveCounter += 1
1082                    else: loadActiveCounter = 0
1083
1084                    if not spriteReloaded and
     loadActiveCounter >= spriteReloadTrigger:
1085                        DexInfo.LoadSpritesheet()
1086                        spriteReloaded = True
1087
1088                    idleCtr += 1
1089                    if click[0] == 1: idleCtr = 0
```

```
1090                    if idleCtr > 1000:
1091                        idleCtr = 0
1092                        DexInfo.SleepState()
1093                        DexInfo.oneTimeCycleLoad = True
1094                        DexInfo.loadNewPokemon = True
1095
1096            DexInfo.running = True
1097            return  DexInfo.currentPokemon
1098
1099        def SleepState():
1100            # PyGame Initialisation
1101            clock = pygame.time.Clock()
1102
1103            # Window and Surface Initialisation
1104            displayWidth = 800
1105            displayHeight = 480
1106
1107            try:
1108                if os.uname()[1] == 'raspberrypi':
1109                    mainSurface = pygame.display.
    set_mode((0,0),pygame.FULLSCREEN)
1110                    pygame.mouse.set_cursor((8,8),(0,0
    ),(0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,0))
1111                else:
1112                    mainSurface = pygame.display.
    set_mode((displayWidth,displayHeight))
1113                    pygame.mouse.set_visible(True)
1114            except:
1115                mainSurface = pygame.display.set_mode((
    displayWidth,displayHeight))
1116                pygame.mouse.set_visible(True)
1117
1118            run = True
1119
1120            pygame.draw.rect(mainSurface,(20,20,20),(0,
    0,800,480))
1121            sleepSurface = pygame.Surface((600,300)).
    convert_alpha()
1122            sleepImg = pygame.image.load("sleeping.png"
    ).convert_alpha()
1123
1124            runtimeCtr = 0
1125
1126            while run:
```

```
1127                for event in pygame.event.get():
1128                    if event.type == pygame.QUIT:
1129                        pygame.quit()
1130                        sys.exit()
1131                click = pygame.mouse.get_pressed()
1132
1133                if click[0] == 1:
1134                    run = False
1135
1136                sleepSurface.fill((20,20,20))
1137                sleepSurface.set_colorkey((0,0,0))
1138                if runtimeCtr % 2 == 0: sleepSurface.
     blit(sleepImg,(0,0))
1139                else: sleepSurface.blit(sleepImg,(-600,
     0))
1140                mainSurface.blit(sleepSurface,(100,180
     ))
1141                Text.Write(mainSurface,(400,140),"
     Sleeping...",30,"joy.otf",(200,200,200),True)
1142                pygame.display.update()
1143
1144                runtimeCtr += 1
1145                if runtimeCtr > 100: runtimeCtr = 0
1146
1147                clock.tick(2)
1148            return
1149
1150
```

```
 1 # Importing Modules
 2 import pygame
 3 from pygame import gfxdraw
 4 from pygame.locals import *
 5 from threading import Thread
 6 import time
 7 import random
 8 import sys
 9 import sqlite3
10 import os
11 import math
12
13 from CButton import Button
14 from SpriteManager import Sprite
15 from CDrawing import Draw
16 from CText import Text
17 from CUserInterface import UI
18
19 from DexInfo import DexInfo
20 from DexSearch import DexSearch
21
22
23 from io import StringIO
24
25 class DexMenu:
26
27 ##########################################################
   ####################################
28 #   PROTECTED VARIABLES

             #
29 ##########################################################
   ####################################
30
31     conn = sqlite3.connect('pokemon.db')
32     conn.row_factory = sqlite3.Row
33     c = conn.cursor()
34
35     thread = Thread()
36     sleepThread = Thread()
37
38     running = True
39     reload = True
40
```

```
41      dexScrollOffset = 0
42
43      searchEnabled = False
44
45  ##########################################################
    ######################################
46  #    FUNCTIONS

                            #
47  ##########################################################
    ######################################
48
49      def GetPokeDataAll():
50          DexInfo.c.execute("""SELECT *,
51                  evoNext.evoNextDex AS nextEvolution,
52                  evoPrev.evoDex AS prevEvolution,
53                  typeA.typeName AS type1Name,
54                  typeB.typeName AS type2Name
55                  FROM pokemon
56                  LEFT JOIN sprites ON pokemon.
    nationalDex = sprites.nationalDex
57                  AND (sprites.isMegaEvolution IS NULL
    OR sprites.isMegaEvolution = '')
58                  LEFT JOIN types AS typeA ON pokemon.
    typeID1 = typeA.id
59                  LEFT JOIN types AS typeB ON pokemon.
    typeID2 = typeB.id
60                  LEFT JOIN regions ON pokemon.regionID
     = regions.id
61                  LEFT JOIN evYields ON pokemon.
    nationalDex = evYields.nationalDex
62                  LEFT JOIN evYieldTypes ON evYields.
    evYieldTypeID = evYieldTypes.id
63                  LEFT JOIN growthRates ON pokemon.
    growthRateID = growthRates.id
64                  LEFT JOIN eggGroups ON pokemon.
    eggGroupID = eggGroups.id
65                  LEFT JOIN evolutions AS evoNext ON
    pokemon.nationalDex = evoNext.evoDex
66                  LEFT JOIN evolutions AS evoPrev ON
    pokemon.nationalDex = evoPrev.evoNextDex
67                  WHERE pokemon.nationalDex IS NOT NULL
68                  ORDER BY pokemon.nationalDex ASC
69                  """)
```

```
 70          return DexInfo.c.fetchall()
 71
 72      def GetGenerations():
 73          DexInfo.c.execute("SELECT * FROM generations
     ORDER BY genNr ASC")
 74          return DexInfo.c.fetchall()
 75
 76      def GetPMGen(dexNr):
 77          parameters = (dexNr,dexNr,)
 78          DexInfo.c.execute("SELECT * FROM generations
     WHERE genDexStart <= ? AND genDexEnd >= ?",
    parameters)
 79          return DexInfo.c.fetchone()["genNr"]
 80
 81      def LoadSpriteSheet(dexSurface,generationData):
 82
 83              yOffset = 0
 84              #cor de fundo da lista dex
 85              dexSurface.fill((40,40,40))
 86              dexSurface.set_colorkey((0,0,0))
 87
 88              for gen in generationData:
 89                  yOffset += 20
 90
 91                  pygame.gfxdraw.aacircle(dexSurface,
    300-100,yOffset,10,(255,255,255))
 92                  pygame.gfxdraw.filled_circle(
    dexSurface,300-100,yOffset,10,(255,255,255))
 93                  pygame.gfxdraw.aacircle(dexSurface,
    300+100,yOffset,10,(255,255,255))
 94                  pygame.gfxdraw.filled_circle(
    dexSurface,300+100,yOffset,10,(255,255,255))
 95 #barra de geracao
 96                  pygame.draw.rect(dexSurface,(255,255
    ,255),(0,yOffset-3,600,7))
 97                  pygame.draw.rect(dexSurface,(255,255
    ,255),(200,yOffset-10,200,21))
 98 #barra de geracao cor da fonte
 99                  Text.Write(dexSurface,(300,yOffset),
    gen["genName"],25,"joy.otf",(255,0,0),True)
100                  yOffset += 20
101
102                  for pm in range(int(gen["genDexStart
    "]),int(gen["genDexEnd"]+1),6):
```

```
103                     for column in range(0,6):
104                             if pm <= int(gen["genDexEnd"]):
105                                 filePath = "sprites/" + str('{0:03d}'.format(pm)) + "/sprite-small-FN-" + str('{0:03d}'.format(pm)) + ".png"
106                                 if os.path.isfile(filePath): pokeSprite = pygame.image.load(filePath).convert_alpha()
107                                 else: pokeSprite = pygame.image.load("notFound.gif").convert_alpha()
108                                 pokeSprite = pygame.transform.scale(pokeSprite,(96,96))
109                                 dexSurface.blit(pokeSprite,(column * 96, yOffset))
110                                 pm += 1
111                     yOffset += 96
112
113 ##########################################################################################
114 #   TOGGLE FUNCTIONS

                    #
115 ##########################################################################################
116
117     def ReturnToMenu():
118         DexMenu.running = False
119         DexMenu.reload = True
120
121     def ToggleGen1():
122         DexMenu.dexScrollOffset = 0
123
124     def ToggleGen2():
125         DexMenu.dexScrollOffset = -(40+26*96)
126
127     def ToggleGen3():
128         DexMenu.dexScrollOffset = - (40+26*96) - (40+17*96)
129
130     def ToggleGen4():
131         DexMenu.dexScrollOffset = - (40+26*96) - (40+17*96) - (40+23*96)
132
```

```
133     def ToggleGen5():
134         DexMenu.dexScrollOffset = - (40+26*96) - (40
    +17*96) - (40+23*96) - (40+18*96)
135 
136     def ToggleGen6():
137         DexMenu.dexScrollOffset = - (40+26*96) - (40
    +17*96) - (40+23*96) - (40+18*96) - (40+26*96)
138 
139     def ToggleGen7():
140         DexMenu.dexScrollOffset = - (40+26*96) - (40
    +17*96) - (40+23*96) - (40+18*96) - (40+26*96) - (40
    +12*96)
141 
142     def ToggleGen8():
143         DexMenu.dexScrollOffset = - (40+26*96) - (40
    +17*96) - (40+23*96) - (40+18*96) - (40+26*96) - (40
    +12*96) - (40+14*96)
144 
145     def ToggleSearch():
146         DexMenu.searchEnabled = True
147 
148 ##################################################
    ########################################
149 ##################################################
    ########################################
150 #   MAIN START

                          #
151 ##################################################
    ########################################
152 ##################################################
    ########################################
153 
154     def Show():
155 
156 ##################################################
    ########################################
157 #   INITIALISATION AND SETUP

          #
158 ##################################################
    ########################################
159 
160         # PyGame Initialisation
```

```
161         clock = pygame.time.Clock()
162 
163         # Window and Surface Initialisation
164         displayWidth = 800
165         displayHeight = 480
166 
167         idleCtr = 0
168 
169         flags = FULLSCREEN | DOUBLEBUF
170 
171         try:
172             if os.uname()[1] == 'raspberrypi':
173                 mainSurface = pygame.display.
    set_mode((0,0),flags)
174                 pygame.mouse.set_cursor((8,8),(0,0
    ),(0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,0))
175             else:
176                 mainSurface = pygame.display.
    set_mode((displayWidth,displayHeight))
177                 pygame.mouse.set_visible(True)
178         except:
179             mainSurface = pygame.display.set_mode((
    displayWidth,displayHeight))
180             pygame.mouse.set_visible(True)
181 
182         pygame.event.set_allowed([QUIT, KEYDOWN,
    KEYUP])
183 
184 ##################################################
    ######################################
185 #   SURFACE DEFINITIONS

                #
186 ##################################################
    ######################################
187 
188 
189         dexSurface = pygame.Surface((600,15000)).
    convert_alpha()
190 
191 ##################################################
    ######################################
192 #   VARIABLE DEFINITIONS
```

```
192             #
193 ##########################################################
    ######################################
194
195         selectionEngaged = False
196         engagedMousePos = (0,0)
197
198         clickCtr = 0
199         scrollCooldown = 0
200
201         scrollDecayEngaged = False
202         scrDecFirstValue = (0,0)
203         scrDecSecondValue = (0,0)
204         scrDecCounter = 0
205         scrollDecay = 0
206         scrollDirectionUp = False
207
208         selectClickEngaged = False
209         passNextEngage = False
210
211         passNextEngageCtr = 0
212
213         mouseDexNr = 0
214
215         idleCtr = 0
216
217         pokeData = DexMenu.GetPokeDataAll()
218         generationData = DexMenu.GetGenerations()
219
220 ##########################################################
    ######################################
221 #   tela de carregamento

                #
222 ##########################################################
    ######################################
223
224         DexMenu.thread = Thread(target = DexMenu.
    LoadSpriteSheet, args = (dexSurface,generationData
    ,))
225         DexMenu.thread.start()
226
227
228
```

```
229         while DexMenu.thread.isAlive():
230             pygame.draw.rect(mainSurface,(0,0,0),(0,
    0,800,480))
231             Text.Write(mainSurface,(400,240),"
    Carregando...",35,"joy.otf",(220,220,220),True)
232             pygame.display.update()
233             clock.tick(30)
234 
235 
236 
237 
238         pygame.image.save(dexSurface,"DexSurface.png
    ");
239 
240 
241 
242 
243         while DexMenu.running:
244 
245 
246 
247             #cor do botao de geracao
248             Button.idleColor = (60,60,60)
249             #cor do botao selecionador
250             Button.hoverColor = (255,0,0)
251             #cor da fonte
252             Button.fontColor = (255,255,255)
253             Button.disabledColor = (150,150,150)
254             #contorno do botao
255             Button.borderColor = (255,0,0)
256             Button.fontFamily = "joy.otf"
257 
258              # Buttons Declarations
259             btnToggleGen1 = Button.RoundRect(
    mainSurface,(5,90,65,50),15,"Ger. 1",18,1,DexMenu.
    ToggleGen1)
260             btnToggleGen2 = Button.RoundRect(
    mainSurface,(75,90,65,50),15,"Ger. 2",18,1,DexMenu.
    ToggleGen2)
261             btnToggleGen3 = Button.RoundRect(
    mainSurface,(5,150,65,50),15,"Ger. 3",18,1,DexMenu.
    ToggleGen3)
262             btnToggleGen4 = Button.RoundRect(
    mainSurface,(75,150,65,50),15,"Ger. 4",18,1,DexMenu.
```

```
262 ToggleGen4)
263             btnToggleGen5 = Button.RoundRect(
    mainSurface,(5,210,65,50),15,"Ger. 5",18,1,DexMenu.
    ToggleGen5)
264             btnToggleGen6 = Button.RoundRect(
    mainSurface,(75,210,65,50),15,"Ger. 6",18,1,DexMenu.
    ToggleGen6)
265             btnToggleGen7 = Button.RoundRect(
    mainSurface,(5,270,65,50),15,"Ger. 7",18,1,DexMenu.
    ToggleGen7)
266             btnToggleGen8 = Button.RoundRect(
    mainSurface,(75,270,65,50),15,"Ger. 8",18,1,DexMenu.
    ToggleGen8)
267             btnSearch = Button.RoundRect(mainSurface
    ,(5,380,135,50),15,"Buscar",18,1,DexMenu.
    ToggleSearch)
268 
269             btnBackButton = Button.RoundRect(
    mainSurface,(10,10,90,60),15,"< Voltar",18,1,DexMenu
    .ReturnToMenu)
270 
271             # One-Time Drawing routines
272             mainSurface.fill((30,30,30))
273             Draw.RoundRect(mainSurface,(40,40,40),(
    150,20,640,410),20,2,(255,0,0),"Pokedex")
274             Draw.Pokeball(mainSurface,(140,35),(255,
    255,255),(40,40,40))
275 
276             pygame.display.update()
277 
278             DexMenu.reload = False
279 
280 ##########################################################
    ######################################
281 #   RUNNING LOOP

                        #
282 ##########################################################
    ######################################
283 
284             while not DexMenu.reload:
285 
286                 # Event Processing
287                 for event in pygame.event.get():
```

```
288                     if event.type == pygame.QUIT:
289                         pygame.quit()
290                         sys.exit()
291 
292                 # Keypress Processing
293                 keys = pygame.key.get_pressed()
294                 if keys[pygame.K_q] != 0:
295                     pygame.quit()
296                     sys.exit()
297 
298                 # Mouse Data
299                 mouse = pygame.mouse.get_pos()
300                 mouseRel = pygame.mouse.get_rel()
301                 click = pygame.mouse.get_pressed()
302 
303                 #laterais dos botoes
304                 mainSurface.fill((30,30,30))
305                 mainSurface.blit(dexSurface,(170,50+
    DexMenu.dexScrollOffset))
306 
307                 #instalacao de botoes de geracao
308                 pygame.display.update(btnToggleGen1.
    Show())
309                 pygame.display.update(btnToggleGen2.
    Show())
310                 pygame.display.update(btnToggleGen3.
    Show())
311                 pygame.display.update(btnToggleGen4.
    Show())
312                 pygame.display.update(btnToggleGen5.
    Show())
313                 pygame.display.update(btnToggleGen6.
    Show())
314                 pygame.display.update(btnToggleGen7.
    Show())
315                 pygame.display.update(btnToggleGen8.
    Show())
316                 pygame.display.update(btnSearch.Show
    ())
317 
318                 pygame.display.update(btnBackButton.
    Show())
319 
320                 # Calculate Dex-Screen Values
```

```
321                 dexOffsetStep = abs(int(DexMenu.
    dexScrollOffset / 96))
322                 if mouseDexNr != None: Draw.
    DexCursor(mainSurface,(mouse[0],mouse[1]),"#" + str(
    '{0:03d}'.format(mouseDexNr)))
323                 else: Draw.DexCursor(mainSurface,(
    mouse[0],mouse[1]))
324                 mouseDexPosRow = abs(DexMenu.
    dexScrollOffset) + mouse[1] - 50-40
325                 mouseDexPosRowOffset = 0
326                 mouseDexOffset = 0
327                 mouseDexPosRowOrig = mouseDexPosRow
328
329                 # Calculating the horizontal and
    vertical offset for each generation
330                 # Every generation 40 are added to
    compensate for the generation-banner
331                 # vertical offset is the "empty
    spaces" at the end of each generation added to the
    next one to
332                 # correct the dex number.
333                 # Calc: DexRows*96 and (DexRows*96)+
    40
334                 if 0 < mouseDexPosRow <= 2496:
335                     mouseDexPosRowOffset -= 0
336                     mouseDexOffset = 0
337                 elif 2496 + 40 < mouseDexPosRow <=
    4168:
338                     mouseDexPosRowOffset -= 40
339                     mouseDexOffset = 5
340                 elif 4168 + 40 < mouseDexPosRow <=
    6416:
341                     mouseDexPosRowOffset -= 80
342                     mouseDexOffset = 7
343                 elif 6416 + 40 < mouseDexPosRow <=
    8184:
344                     mouseDexPosRowOffset -= 120
345                     mouseDexOffset = 10
346                 elif 8184 + 40 < mouseDexPosRow <=
    10720:
347                     mouseDexPosRowOffset -= 160
348                     mouseDexOffset = 11
349                 elif 10720 + 40 < mouseDexPosRow <=
    11912:
```

```
350                     mouseDexPosRowOffset -= 200
351                     mouseDexOffset = 11
352                 elif 11912 + 40 < mouseDexPosRow <=
    13296:
353                     mouseDexPosRowOffset -= 240
354                     mouseDexOffset = 11
355                 elif 13296 + 40 < mouseDexPosRow <=
    99999:
356                     mouseDexPosRowOffset -= 280
357                     mouseDexOffset = 13
358                 else:
359                     mouseDexPosRowOffset = 0
360 
361                 # Additional calculations for Dex-
    Screen
362                 mouseDexPosRow +=
    mouseDexPosRowOffset
363                 mouseDexRow = math.ceil(
    mouseDexPosRow/96)-1
364                 mouseDexPosCol = mouse[0]-170
365                 mouseDexCol = math.ceil(
    mouseDexPosCol/96) - 1
366                 mouseDexNr = (mouseDexRow*6 +
    mouseDexCol)+1 - mouseDexOffset
367 
368                 # Checking if the mouseDexNr is
    valid
369                 # Gets set to None when over an "
    empty field" for exsample
370                 if mouse[0] < 170: mouseDexNr = None
371                 elif 0 < mouseDexPosRow <= 2496 and
    generationData[0]["genDexStart"] <= mouseDexNr <=
    generationData[0]["genDexEnd"]: pass
372                 elif 2496 + 40 < mouseDexPosRowOrig
     <= 4168 and generationData[1]["genDexStart"] <=
    mouseDexNr <= generationData[1]["genDexEnd"]: pass
373                 elif 4168 + 40 < mouseDexPosRowOrig
     <= 6416 and generationData[2]["genDexStart"] <=
    mouseDexNr <= generationData[2]["genDexEnd"]: pass
374                 elif 6416 + 40 < mouseDexPosRowOrig
     <= 8184 and generationData[3]["genDexStart"] <=
    mouseDexNr <= generationData[3]["genDexEnd"]: pass
375                 elif 8184 + 40 < mouseDexPosRowOrig
     <= 10720 and generationData[4]["genDexStart"] <=
```

```
375 mouseDexNr <= generationData[4]["genDexEnd"]: pass
376                 elif 10720 + 40 < mouseDexPosRowOrig
     <= 11912 and generationData[5]["genDexStart"] <=
    mouseDexNr <= generationData[5]["genDexEnd"]: pass
377                 elif 11912 + 40 < mouseDexPosRowOrig
     <= 13296 and generationData[6]["genDexStart"] <=
    mouseDexNr <= generationData[6]["genDexEnd"]: pass
378                 elif 13296 + 40 < mouseDexPosRowOrig
     <= 99999 and generationData[7]["genDexStart"] <=
    mouseDexNr <= generationData[7]["genDexEnd"]: pass
379                 else:  mouseDexNr = None
380
381
382                 # Click general
383                 if selectClickEngaged and
    passNextEngageCtr <= 0:
384                     if click[0] == 0:
385                         if selecteEngagedPos[0]-10
     < mouse[0] < selecteEngagedPos[0]+10 and
    selecteEngagedPos[1]-10 < mouse[1] <
    selecteEngagedPos[1]+10:
386                             if mouseDexNr != None:
387                                 selectClickEngaged
     = False
388                                 DexMenu.reload =
    True
389                                 passNextEngageCtr =
    10
390                                 selectedDex =
    DexInfo.Show(mouseDexNr)
391                                 #DexMenu.
    dexScrollOffset = -((DexMenu.GetPMGen(selectedDex)-1
    )*40 + ((selectedDex+2*DexMenu.GetPMGen(selectedDex
    )/6)*96) - (2*96))
392                             selectClickEngaged = False
393
394                 if click[0] == 1 and not
    selectClickEngaged and passNextEngageCtr <= 0:
395                     selectClickEngaged = True
396                     selecteEngagedPos = mouse
397                 passNextEngage = False
398                 if passNextEngageCtr > 0:
    passNextEngageCtr -= 1
399
```

```
400                 # Scrolling generall
401                 if click[0] == 1: clickCtr += 1
402                 else: clickCtr = 0
403 
404                 if click[0] == 1 and clickCtr > 1:
405                     DexMenu.dexScrollOffset += 2*
    mouseRel[1]
406                     if DexMenu.dexScrollOffset > 0:
    DexMenu.dexScrollOffset = 0
407                     if DexMenu.dexScrollOffset < -(
    802/6)*96 - 3*96: DexMenu.dexScrollOffset = -(802/6
    )*96 - 3*96
408 
409 
410                 if DexMenu.searchEnabled:
411                     selectedDex = DexSearch.Show()
412                     DexMenu.reload = True
413                     passNextEngageCtr = 10
414                     if selectedDex != None:
    selectedDex = DexInfo.Show(selectedDex)
415                     DexMenu.searchEnabled = False
416 
417 
418 
419                 # Descanso de tela
420                 idleCtr += 1
421                 if click[0] == 1: idleCtr = 0
422                 #tempo para descanso
423                 if idleCtr > 1000:
424                     idleCtr = 0
425                     DexMenu.SleepState()
426                     DexMenu.reload = True
427 
428                 # Update Screen
429                 pygame.display.update((170,50,600,
    380))
430                 clock.tick(60)
431 
432         DexMenu.running = True
433         return
434 
435     def SleepState():
436         # PyGame Initialisation
437         clock = pygame.time.Clock()
```

```
438
439         # Window and Surface Initialisation
440         displayWidth = 800
441         displayHeight = 480
442
443         try:
444             if os.uname()[1] == 'raspberrypi':
445                 mainSurface = pygame.display.
    set_mode((0,0),pygame.FULLSCREEN)
446                 pygame.mouse.set_cursor((8,8),(0,0
    ),(0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,0))
447             else:
448                 mainSurface = pygame.display.
    set_mode((displayWidth,displayHeight))
449                 pygame.mouse.set_visible(True)
450         except:
451             mainSurface = pygame.display.set_mode((
    displayWidth,displayHeight))
452             pygame.mouse.set_visible(True)
453
454         run = True
455
456         pygame.draw.rect(mainSurface,(20,20,20),(0,0
    ,800,480))
457         sleepSurface = pygame.Surface((600,300)).
    convert_alpha()
458         sleepImg = pygame.image.load("sleeping.png"
    ).convert_alpha()
459
460         runtimeCtr = 0
461
462         while run:
463             for event in pygame.event.get():
464                 if event.type == pygame.QUIT:
465                     pygame.quit()
466                     sys.exit()
467             click = pygame.mouse.get_pressed()
468
469             if click[0] == 1:
470                 run = False
471
472             sleepSurface.fill((20,20,20))
473             sleepSurface.set_colorkey((0,0,0))
474             if runtimeCtr % 2 == 0: sleepSurface.
```

```
474 blit(sleepImg,(0,0))
475             else: sleepSurface.blit(sleepImg,(-600,0
    ))
476             mainSurface.blit(sleepSurface,(100,180))
477             Text.Write(mainSurface,(400,140),"
    Deslize o dedo para desbloquear",30,"joy.otf",(200,
    200,200),True)
478             pygame.display.update()
479 
480             runtimeCtr += 1
481             if runtimeCtr > 100: runtimeCtr = 0
482 
483             clock.tick(2)
484         return
```

```
# -*- mode: python -*-

block_cipher = None


a = Analysis(['PiDex.py'],
             pathex=['S:\\VisualStudioProjects\\PiDex\\Assembled_Ver2\\PiDex\\PiDex'],
             binaries=[],
             datas=[],
             hiddenimports=[],
             hookspath=[],
             runtime_hooks=[],
             excludes=[],
             win_no_prefer_redirects=False,
             win_private_assemblies=False,
             cipher=block_cipher,
             noarchive=False)
pyz = PYZ(a.pure, a.zipped_data,
             cipher=block_cipher)
exe = EXE(pyz,
          a.scripts,
          a.binaries,
          a.zipfiles,
          a.datas,
          [],
          name='PiDex',
          debug=False,
          bootloader_ignore_signals=False,
          strip=False,
          upx=True,
          runtime_tmpdir=None,
          console=False )

```

```
 1 import pygame
 2 from pygame import gfxdraw
 3 from CText import Text
 4
 5 class Draw(object):
 6     """description of class"""
 7
 8     def RoundRect(surface,color,rect,radius,
   borderWidth=None,borderColor=(0,0,0),titlebar = None
   ):
 9
10         if borderWidth != None:  Draw.RoundRect(
   surface,borderColor,(rect[0]-borderWidth,rect[1]-
   borderWidth,rect[2]+2*borderWidth,rect[3]+2*
   borderWidth),radius+borderWidth)
11
12         pygame.gfxdraw.aacircle(surface,rect[0]+
   radius,rect[1]+radius,radius,color)
13         pygame.gfxdraw.filled_circle(surface,rect[0]+
   radius,rect[1]+radius,radius,color)
14         pygame.gfxdraw.aacircle(surface,rect[0]-
   radius + rect[2]-1,rect[1]+radius,radius,color)
15         pygame.gfxdraw.filled_circle(surface,rect[0]-
   radius + rect[2]-1,rect[1]+radius,radius,color)
16         pygame.gfxdraw.aacircle(surface,rect[0]+
   radius,rect[1]-radius + rect[3]-1,radius,color)
17         pygame.gfxdraw.filled_circle(surface,rect[0]+
   radius,rect[1]-radius + rect[3]-1,radius,color)
18         pygame.gfxdraw.aacircle(surface,rect[0]-
   radius + rect[2]-1,rect[1]-radius + rect[3]-1,radius,
   color)
19         pygame.gfxdraw.filled_circle(surface,rect[0]-
   radius + rect[2]-1,rect[1]-radius + rect[3]-1,radius,
   color)
20
21         pygame.draw.rect(surface,color,(rect[0] +
   radius, rect[1], rect[2] - 2*radius, rect[3]))
22         pygame.draw.rect(surface,color,(rect[0], rect
   [1] + radius, rect[2], rect[3] - 2*radius))
23
24         if titlebar != None:
25             pygame.gfxdraw.aacircle(surface,rect[0]+
   radius,rect[1]+radius,radius,borderColor)
26             pygame.gfxdraw.filled_circle(surface,rect
```

```
26 [0]+radius,rect[1]+radius,radius,borderColor)
27             pygame.gfxdraw.aacircle(surface,rect[0]+
   rect[2]-radius,rect[1]+radius,radius,borderColor)
28             pygame.gfxdraw.filled_circle(surface,rect
   [0]+rect[2]-radius,rect[1]+radius,radius,borderColor)
29 
30             pygame.draw.rect(surface,borderColor,(
   rect[0]+radius,rect[1],rect[2]-2*radius,radius))
31             pygame.draw.rect(surface,color,(rect[0],
   rect[1]+radius,rect[2],radius+1))
32             pygame.draw.rect(surface,borderColor,(
   rect[0],rect[1]+radius,rect[2],10))
33             Text.Write(surface,(rect[0]+(rect[2]/2),
   rect[1]+14),titlebar,20,"joy.otf",(255,255,255),True)
34 
35         if borderWidth != None: return (rect[0]-
   borderWidth,rect[1]-borderWidth,rect[2]+2*borderWidth
   ,rect[3]+2*borderWidth)
36         else: return rect
37 
38     def Circle(surface,color,pos,radius,borderWidth=
   None,borderColor=(0,0,0)):
39         if borderWidth != None:  Draw.Circle(surface,
   borderColor,pos,radius+borderWidth)
40 
41         pygame.gfxdraw.aacircle(surface,pos[0],pos[1
   ],radius,color)
42         pygame.gfxdraw.filled_circle(surface,pos[0],
   pos[1],radius,color)
43 
44         if borderWidth == None: return (pos[0]-radius
   ,pos[1]-radius,2*radius+2,2*radius+2)
45         else: return (pos[0]-radius-borderWidth,pos[1
   ]-radius-borderWidth,2*radius+2*borderWidth+2,2*
   radius+2*borderWidth+2)
46 
47     def Pokeball(surface,pos,color,backcolor):
48         pygame.gfxdraw.aacircle(surface,pos[0],pos[1
   ],32,color)
49         pygame.gfxdraw.filled_circle(surface,pos[0],
   pos[1],32,color)
50 
51         pygame.gfxdraw.aacircle(surface,pos[0],pos[1
   ],29,backcolor)
```

```
52          pygame.gfxdraw.filled_circle(surface,pos[0],
    pos[1],29,backcolor)
53
54          pygame.draw.rect(surface,color,(pos[0]-30,pos
    [1]-1,31*2,3))
55
56          pygame.gfxdraw.aacircle(surface,pos[0],pos[1
    ],12,color)
57          pygame.gfxdraw.filled_circle(surface,pos[0],
    pos[1],12,color)
58
59          pygame.gfxdraw.aacircle(surface,pos[0],pos[1
    ],10,backcolor)
60          pygame.gfxdraw.filled_circle(surface,pos[0],
    pos[1],10,backcolor)
61
62          pygame.gfxdraw.aacircle(surface,pos[0],pos[1
    ],5,color)
63          pygame.gfxdraw.filled_circle(surface,pos[0],
    pos[1],5,color)
64
65          return pos
66
67      def TypeSign1(surface,pos,color,text):
68          pygame.gfxdraw.aapolygon(surface,((pos[0],pos
    [1]),(pos[0]+145,pos[1]),(pos[0]+115,pos[1]+35),(pos[
    0],pos[1]+35),(pos[0],pos[1])),color)
69          pygame.gfxdraw.filled_polygon(surface,((pos[0
    ],pos[1]),(pos[0]+145,pos[1]),(pos[0]+115,pos[1]+35
    ),(pos[0],pos[1]+35),(pos[0],pos[1])),color)
70          Text.Write(surface,(pos[0]+70,pos[1]+18),text
    ,20,"joy.otf",(255,255,255),True)
71          return (pos[0],pos[1],145,35)
72
73      def TypeSign2(surface,pos,color,text):
74          pygame.gfxdraw.aapolygon(surface,((pos[0]+30,
    pos[1]),(pos[0]+145,pos[1]),(pos[0]+145,pos[1]+35),(
    pos[0],pos[1]+35),(pos[0]+30,pos[1])),color)
75          pygame.gfxdraw.filled_polygon(surface,((pos[0
    ]+30,pos[1]),(pos[0]+145,pos[1]),(pos[0]+145,pos[1]+
    35),(pos[0],pos[1]+35),(pos[0]+30,pos[1])),color)
76          Text.Write(surface,(pos[0]+75,pos[1]+18),text
    ,20,"joy.otf",(255,255,255),True)
77          return (pos[0],pos[1],145,35)
```

```
78
79      def TypeSignSingle(surface,pos,color,text):
80          pygame.gfxdraw.aapolygon(surface,((pos[0],
    pos[1]),(pos[0]+220,pos[1]),(pos[0]+190,pos[1]+35),(
    pos[0],pos[1]+35),(pos[0],pos[1])),color)
81          pygame.gfxdraw.filled_polygon(surface,((pos[
    0],pos[1]),(pos[0]+220,pos[1]),(pos[0]+190,pos[1]+35
    ),(pos[0],pos[1]+35),(pos[0],pos[1])),color)
82          pygame.gfxdraw.aapolygon(surface,((pos[0]+30
    +200,pos[1]),(pos[0]+200+70,pos[1]),(pos[0]+200+70,
    pos[1]+35),(pos[0]+200,pos[1]+35),(pos[0]+30+200,pos
    [1])),color)
83          pygame.gfxdraw.filled_polygon(surface,((pos[
    0]+30+200,pos[1]),(pos[0]+200+70,pos[1]),(pos[0]+200
    +70,pos[1]+35),(pos[0]+200,pos[1]+35),(pos[0]+30+200
    ,pos[1])),color)
84          Text.Write(surface,(pos[0]+75,pos[1]+18),
    text,20,"joy.otf",(255,255,255),True)
85          return (pos[0],pos[1],220,35)
86
87      def RRCursor(surface,color,rect,radius,
    borderWidth=None,borderColor=(0,0,0)):
88
89          if borderWidth != None:  Draw.RRCursor(
    surface,borderColor,(rect[0]-borderWidth,rect[1]-
    borderWidth,rect[2]+2*borderWidth,rect[3]+2*
    borderWidth),radius+borderWidth)
90
91          pygame.gfxdraw.aacircle(surface,rect[0]+
    radius,rect[1]+radius,radius,color)
92          pygame.gfxdraw.filled_circle(surface,rect[0
    ]+radius,rect[1]+radius,radius,color)
93
94
95          pygame.gfxdraw.aacircle(surface,rect[0]-
    radius + rect[2]-1,rect[1]-radius + rect[3]-1,radius
    ,color)
96          pygame.gfxdraw.filled_circle(surface,rect[0
    ]-radius + rect[2]-1,rect[1]-radius + rect[3]-1,
    radius,color)
97
98          pygame.draw.rect(surface,color,(rect[0] +
    radius, rect[1], rect[2] - 2*radius, rect[3]))
99          pygame.draw.rect(surface,color,(rect[0],
```

```
 99 rect[1] + radius, rect[2], rect[3] - 2*radius))
100
101
102         if borderWidth != None: return (rect[0]-
    borderWidth,rect[1]-borderWidth,rect[2]+2*
    borderWidth,rect[3]+2*borderWidth)
103         else: return rect
104
105     def DexCursor(surface, pos, text = None):
106         b1 = ( (pos[0],pos[1]-48) , (pos[0]-40,pos[1
    ]-48) , (pos[0]-48,pos[1]-40) , (pos[0]-48,pos[1]) )
107         b2 = ( (pos[0],pos[1]+48) , (pos[0]+40,pos[1
    ]+48) , (pos[0]+48,pos[1]+40) , (pos[0]+48,pos[1]) )
108
109         if text != None: Text.Write(surface,(pos[0]-
    40,pos[1]-40),text,18,"joy.otf",(255,255,255))
110
111         pygame.draw.lines(surface,(255,0,0),False,b1
    ,5)
112         pygame.draw.lines(surface,(255,255,255),
    False,b2,5)
113
114     def AAFilledCircle(surface,x,y,r,infillColor,
    borderColor,borderWidth=None):
115
116         pygame.gfxdraw.aacircle(surface,x,y,r,
    borderColor)
117         pygame.gfxdraw.filled_circle(surface,x,y,r,
    borderColor)
118
119         if borderWidth != None:
120             pygame.gfxdraw.aacircle(surface,x,y,r-
    borderWidth,infillColor)
121             pygame.gfxdraw.filled_circle(surface,x,y
    ,r-borderWidth,infillColor)
122
123     class Arrow:
124
125         def Left(surface,color,pos,size):
126             pygame.gfxdraw.aapolygon(surface,((pos[0
    ],pos[1]-10*size),(pos[0]+10*size,pos[1]),(pos[0],
    pos[1]+10*size),(pos[0],pos[1]-10*size)),color)
127             pygame.gfxdraw.filled_polygon(surface,((
    pos[0],pos[1]-10*size),(pos[0]+10*size,pos[1]),(pos[
```

```
127 0],pos[1]+10*size),(pos[0],pos[1]-10*size)),color)
128
129
130
```

```python
# Importing Modules
import pygame
from pygame import gfxdraw
from pygame.locals import *
from threading import Thread
import time
import random
import sys
import sqlite3
import os
import math

from CButton import Button
from SpriteManager import Sprite
from CDrawing import Draw
from CText import Text

class DexSearch:

##########################################################################################
#   PROTECTED VARIABLES

            #
##########################################################################################

    conn = sqlite3.connect('pokemon.db')
    conn.row_factory = sqlite3.Row
    c = conn.cursor()

    thread = Thread()
    sleepThread = Thread()

    running = True
    reload = True

    searchVal = ""
    engageSearch = False

##########################################################################################
#   FUNCTIONS
```

```
38                          #
39 ##########################################################
   ####################################
40
41     def PasteLetter(letter):
42         DexSearch.searchVal = DexSearch.searchVal +
   letter
43         DexSearch.engageSearch = True
44
45     def DeleteLetter():
46         DexSearch.searchVal = DexSearch.searchVal[:-1
   ]
47         DexSearch.engageSearch = True
48
49 ##########################################################
   ####################################
50 #   TOGGLE FUNCTIONS

                    #
51 ##########################################################
   ####################################
52
53     def ReturnToMenu():
54         DexSearch.running = False
55         DexSearch.reload = True
56
57 ##########################################################
   ####################################
58 ##########################################################
   ####################################
59 #   MAIN START

                          #
60 ##########################################################
   ####################################
61 ##########################################################
   ####################################
62
63     def Show():
64
65 ##########################################################
   ####################################
66 #   INITIALISATION AND SETUP
```

```
66          #
67 ##########################################################
   ######################################
68
69          # PyGame Initialisation
70          clock = pygame.time.Clock()
71
72          # Window and Surface Initialisation
73          displayWidth = 800
74          displayHeight = 480
75
76          idleCtr = 0
77
78
79
80          flags = FULLSCREEN | DOUBLEBUF
81
82          try:
83              if os.uname()[1] == 'raspberrypi':
84                  mainSurface = pygame.display.
   set_mode((0,0),flags)
85                  pygame.mouse.set_cursor((8,8),(0,0
   ),(0,0,0,0,0,0,0,0),(0,0,0,0,0,0,0,0))
86              else:
87                  mainSurface = pygame.display.
   set_mode((displayWidth,displayHeight))
88                  pygame.mouse.set_visible(True)
89          except:
90              mainSurface = pygame.display.set_mode((
   displayWidth,displayHeight))
91              pygame.mouse.set_visible(True)
92
93          pygame.event.set_allowed([QUIT, KEYDOWN,
   KEYUP])
94
95 ##########################################################
   ######################################
96 #   SURFACE DEFINITIONS

                #
97 ##########################################################
   ######################################
98
99          searchResSurface = pygame.Surface((760,110
```

```
 99 )).convert_alpha()
100
101 ##########################################################
    ####################################
102 #   VARIABLE DEFINITIONS

                #
103 ##########################################################
    ####################################
104
105         searchResult = []
106
107 ##########################################################
    ####################################
108 #   LOADING LOOP

                        #
109 ##########################################################
    ####################################
110
111         while DexSearch.running:
112
113             # configuracoes botoes
114             #cor do botao
115             Button.idleColor = (60,60,60)
116             #cor de selecao
117             Button.hoverColor = (255,0,0)
118             #cor da fonte
119             Button.fontColor = (255,255,255)
120
121             Button.disabledColor = (150,150,150)
122             #borda do botao
123             Button.borderColor = (255,0,0)
124             Button.fontFamily = "joy.otf"
125
126             fontSize = 40
127
128             vertOffset = 160
129
130             # Primeira linha
131             btnLetterQ = Button.RoundRect(
    mainSurface,(14+0*78,vertOffset,70,70),15,"Q",
    fontSize,1,DexSearch.PasteLetter,"Q")
132             btnLetterW = Button.RoundRect(
```

```
132 mainSurface,(14+1*78,vertOffset,70,70),15,"W",
    fontSize,1,DexSearch.PasteLetter,"W")
133             btnLetterE = Button.RoundRect(
    mainSurface,(14+2*78,vertOffset,70,70),15,"E",
    fontSize,1,DexSearch.PasteLetter,"E")
134             btnLetterR = Button.RoundRect(
    mainSurface,(14+3*78,vertOffset,70,70),15,"R",
    fontSize,1,DexSearch.PasteLetter,"R")
135             btnLetterT = Button.RoundRect(
    mainSurface,(14+4*78,vertOffset,70,70),15,"T",
    fontSize,1,DexSearch.PasteLetter,"T")
136             btnLetterY = Button.RoundRect(
    mainSurface,(14+5*78,vertOffset,70,70),15,"Y",
    fontSize,1,DexSearch.PasteLetter,"Y")
137             btnLetterU = Button.RoundRect(
    mainSurface,(14+6*78,vertOffset,70,70),15,"U",
    fontSize,1,DexSearch.PasteLetter,"U")
138             btnLetterI = Button.RoundRect(
    mainSurface,(14+7*78,vertOffset,70,70),15,"I",
    fontSize,1,DexSearch.PasteLetter,"I")
139             btnLetterO = Button.RoundRect(
    mainSurface,(14+8*78,vertOffset,70,70),15,"O",
    fontSize,1,DexSearch.PasteLetter,"O")
140             btnLetterP = Button.RoundRect(
    mainSurface,(14+9*78,vertOffset,70,70),15,"P",
    fontSize,1,DexSearch.PasteLetter,"P")
141             # Segunda linha
142             btnLetterA = Button.RoundRect(
    mainSurface,(45+0*80,vertOffset+80,70,70),15,"A",
    fontSize,1,DexSearch.PasteLetter,"A")
143             btnLetterS = Button.RoundRect(
    mainSurface,(45+1*80,vertOffset+80,70,70),15,"S",
    fontSize,1,DexSearch.PasteLetter,"S")
144             btnLetterD = Button.RoundRect(
    mainSurface,(45+2*80,vertOffset+80,70,70),15,"D",
    fontSize,1,DexSearch.PasteLetter,"D")
145             btnLetterF = Button.RoundRect(
    mainSurface,(45+3*80,vertOffset+80,70,70),15,"F",
    fontSize,1,DexSearch.PasteLetter,"F")
146             btnLetterG = Button.RoundRect(
    mainSurface,(45+4*80,vertOffset+80,70,70),15,"G",
    fontSize,1,DexSearch.PasteLetter,"G")
147             btnLetterH = Button.RoundRect(
    mainSurface,(45+5*80,vertOffset+80,70,70),15,"H",
```

```
147 fontSize,1,DexSearch.PasteLetter,"H")
148             btnLetterJ = Button.RoundRect(
    mainSurface,(45+6*80,vertOffset+80,70,70),15,"J",
    fontSize,1,DexSearch.PasteLetter,"J")
149             btnLetterK = Button.RoundRect(
    mainSurface,(45+7*80,vertOffset+80,70,70),15,"K",
    fontSize,1,DexSearch.PasteLetter,"K")
150             btnLetterL = Button.RoundRect(
    mainSurface,(45+8*80,vertOffset+80,70,70),15,"L",
    fontSize,1,DexSearch.PasteLetter,"L")
151             # Third Row
152             btnLetterZ = Button.RoundRect(
    mainSurface,(85+0*80,vertOffset+160,70,70),15,"Z",
    fontSize,1,DexSearch.PasteLetter,"Z")
153             btnLetterX = Button.RoundRect(
    mainSurface,(85+1*80,vertOffset+160,70,70),15,"X",
    fontSize,1,DexSearch.PasteLetter,"X")
154             btnLetterC = Button.RoundRect(
    mainSurface,(85+2*80,vertOffset+160,70,70),15,"C",
    fontSize,1,DexSearch.PasteLetter,"C")
155             btnLetterV = Button.RoundRect(
    mainSurface,(85+3*80,vertOffset+160,70,70),15,"V",
    fontSize,1,DexSearch.PasteLetter,"V")
156             btnLetterB = Button.RoundRect(
    mainSurface,(85+4*80,vertOffset+160,70,70),15,"B",
    fontSize,1,DexSearch.PasteLetter,"B")
157             btnLetterN = Button.RoundRect(
    mainSurface,(85+5*80,vertOffset+160,70,70),15,"N",
    fontSize,1,DexSearch.PasteLetter,"N")
158             btnLetterM = Button.RoundRect(
    mainSurface,(85+6*80,vertOffset+160,70,70),15,"M",
    fontSize,1,DexSearch.PasteLetter,"M")
159             btnSignDot = Button.RoundRect(
    mainSurface,(85+7*80,vertOffset+160,50,70),15,".",
    fontSize,1,DexSearch.PasteLetter,".")
160             btnSignMin = Button.RoundRect(
    mainSurface,(60+85+7*80,vertOffset+160,50,70),15,"-"
    ,fontSize,1,DexSearch.PasteLetter,"-")
161             # Terceira linha
162             btnOpDelete = Button.RoundRect(
    mainSurface,(35+0*80,vertOffset+240,160,70),15,"
    Deletar",30,1,DexSearch.DeleteLetter)
163             btnSignSpa = Button.RoundRect(
    mainSurface,(45+2*80,vertOffset+240,390,70),15,"",30
```

```
163 ,1,DexSearch.PasteLetter," ")
164             btnOpSearch = Button.RoundRect(
    mainSurface,(45+7*80,vertOffset+240,160,70),15,"
    Buscar",30,1,DexSearch.PasteLetter,"")
165 
166             btnGoBack = Button.RoundRect(mainSurface
    ,(0,0,120,30),10,"< Voltar",20,1,DexSearch.
    ReturnToMenu)
167 
168             searchResSurface.fill((40,40,40))
169             searchResSurface.set_colorkey((0,0,0))
170 
171             DexSearch.searchVal = ""
172 
173             DexSearch.reload = False
174 
175 ##################################################
    ######################################
176 #   RUNNING LOOP

                        #
177 ##################################################
    ######################################
178 
179             while not DexSearch.reload:
180 
181                 # Event Processing
182                 for event in pygame.event.get():
183                     if event.type == pygame.QUIT:
184                         pygame.quit()
185                         sys.exit()
186 
187                 # Keypress Processing
188                 keys = pygame.key.get_pressed()
189                 if keys[pygame.K_q] != 0:
190                     pygame.quit()
191                     sys.exit()
192 
193                 mouse = pygame.mouse.get_pos()
194                 click = pygame.mouse.get_pressed()
195 
196 
197                 mainSurface.fill((30,30,30))
198 
```

```
199
200                 Draw.RoundRect(mainSurface,(40,40,40
    ),(5,150,790,325),15,2,(255,0,0))
201                 Draw.RoundRect(mainSurface,(40,40,40
    ),(5,5,790,135),15,2,(255,0,0),"Buscar por: " + str(
    DexSearch.searchVal))
202
203                 if DexSearch.engageSearch:
204                     params = (DexSearch.searchVal +
    "%",)
205                     DexSearch.c.execute("SELECT *
    FROM pokemon WHERE name LIKE ? ORDER BY nationalDex
    ASC LIMIT 0,8",params)
206                     searchResult = DexSearch.c.
    fetchall()
207
208                     i = 0
209
210                     searchResSurface.fill((40,40,40
    ))
211                     searchResSurface.set_colorkey((0
    ,0,0))
212
213                     for pm in searchResult:
214                         filePath = "sprites/" + str(
    '{0:03d}'.format(pm["nationalDex"])) + "/sprite-
    small-FN-" + str('{0:03d}'.format(pm["nationalDex"
    ])) + ".png"
215                         if os.path.isfile(filePath
    ): pokeSprite = pygame.image.load(filePath).
    convert_alpha()
216                         else: pokeSprite = pygame.
    image.load("notFound.gif").convert_alpha()
217                         pokeSprite = pygame.
    transform.scale(pokeSprite,(96,96))
218                         searchResSurface.blit(
    pokeSprite,(i*96,0))
219                         Text.Write(searchResSurface
    ,(i*96+48,96),pm["name"],12,"joy.otf",(255,255,255),
    True)
220                         Text.Write(searchResSurface
    ,(i*96,0),"#" + str('{0:03d}'.format(pm["nationalDex
    "])),12,"joy.otf",(255,255,255),False)
221                         i+=1
```

```
222
223                     DexSearch.engageSearch = False
224
225                 i = 0
226                 for pm in searchResult:
227
228                     if 20+i*96 < mouse[0] < 20+96+i*
    96 and 30 < mouse[1] < 140:
229                         if click[0] == 1: return pm[
    "nationalDex"]
230                     i += 1
231
232
233
234                 mainSurface.blit(searchResSurface,(
    20,30))
235
236                 pygame.display.update(btnLetterQ.
    Show())
237                 pygame.display.update(btnLetterW.
    Show())
238                 pygame.display.update(btnLetterE.
    Show())
239                 pygame.display.update(btnLetterR.
    Show())
240                 pygame.display.update(btnLetterT.
    Show())
241                 pygame.display.update(btnLetterY.
    Show())
242                 pygame.display.update(btnLetterU.
    Show())
243                 pygame.display.update(btnLetterI.
    Show())
244                 pygame.display.update(btnLetterO.
    Show())
245                 pygame.display.update(btnLetterP.
    Show())
246
247                 pygame.display.update(btnLetterA.
    Show())
248                 pygame.display.update(btnLetterS.
    Show())
249                 pygame.display.update(btnLetterD.
    Show())
```

```
250                 pygame.display.update(btnLetterF.
    Show())
251                 pygame.display.update(btnLetterG.
    Show())
252                 pygame.display.update(btnLetterH.
    Show())
253                 pygame.display.update(btnLetterJ.
    Show())
254                 pygame.display.update(btnLetterK.
    Show())
255                 pygame.display.update(btnLetterL.
    Show())
256
257                 pygame.display.update(btnLetterZ.
    Show())
258                 pygame.display.update(btnLetterX.
    Show())
259                 pygame.display.update(btnLetterC.
    Show())
260                 pygame.display.update(btnLetterV.
    Show())
261                 pygame.display.update(btnLetterB.
    Show())
262                 pygame.display.update(btnLetterN.
    Show())
263                 pygame.display.update(btnLetterM.
    Show())
264                 pygame.display.update(btnSignDot.
    Show())
265                 pygame.display.update(btnSignMin.
    Show())
266
267                 pygame.display.update(btnOpDelete.
    Show())
268                 pygame.display.update(btnSignSpa.
    Show())
269                 pygame.display.update(btnOpSearch.
    Show())
270
271                 pygame.display.update(btnGoBack.Show
    ())
272
273                 if 10 < mouse[0] < 790 and 10 <
    mouse[1] < 140:
```

```
274                         Draw.DexCursor(mainSurface,(
    mouse[0],mouse[1]))
275 
276                     # Update Screen
277                     pygame.display.update()
278                     clock.tick(60)
279 
280             DexSearch.running = True
281             return None
```

```
 1 <Project DefaultTargets="Build" xmlns="http://schemas
   .microsoft.com/developer/msbuild/2003" ToolsVersion="
   4.0">
 2   <PropertyGroup>
 3     <Configuration Condition=" '$(Configuration
   )' == '' ">Debug</Configuration>
 4     <SchemaVersion>2.0</SchemaVersion>
 5     <ProjectGuid>66168095-8e00-4a26-90da-da89710aba41
   </ProjectGuid>
 6     <ProjectHome>.</ProjectHome>
 7     <StartupFile>PiDex.py</StartupFile>
 8     <SearchPath>
 9     </SearchPath>
10     <WorkingDirectory>.</WorkingDirectory>
11     <OutputPath>.</OutputPath>
12     <Name>PiDex</Name>
13     <RootNamespace>PiDex</RootNamespace>
14   </PropertyGroup>
15   <PropertyGroup Condition=" '$(Configuration)' == '
   Debug' ">
16     <DebugSymbols>true</DebugSymbols>
17     <EnableUnmanagedDebugging>false</
   EnableUnmanagedDebugging>
18   </PropertyGroup>
19   <PropertyGroup Condition=" '$(Configuration)' == '
   Release' ">
20     <DebugSymbols>true</DebugSymbols>
21     <EnableUnmanagedDebugging>false</
   EnableUnmanagedDebugging>
22   </PropertyGroup>
23   <ItemGroup>
24     <Compile Include="CButton.py" />
25     <Compile Include="CDrawing.py" />
26     <Compile Include="CText.py" />
27     <Compile Include="CUserInterface.py" />
28     <Compile Include="DexHome.py">
29       <SubType>Code</SubType>
30     </Compile>
31     <Compile Include="DexInfo.py" />
32     <Compile Include="DexMenu.py">
33       <SubType>Code</SubType>
34     </Compile>
35     <Compile Include="DexSearch.py">
36       <SubType>Code</SubType>
```

```
37      </Compile>
38      <Compile Include="PiDex.py" />
39      <Compile Include="SpriteManager.py" />
40    </ItemGroup>
41    <Import Project="$(MSBuildExtensionsPath32)\
   Microsoft\VisualStudio\v$(VisualStudioVersion)\Python
    Tools\Microsoft.PythonTools.targets" />
42    <!-- Uncomment the CoreCompile target to enable the
    Build command in
43         Visual Studio and specify your pre- and post-
   build commands in
44         the BeforeBuild and AfterBuild targets below
   . -->
45    <!--<Target Name="CoreCompile" />-->
46    <Target Name="BeforeBuild">
47    </Target>
48    <Target Name="AfterBuild">
49    </Target>
50 </Project>
```

```
 1 import pygame
 2 import os
 3
 4 class Sprite(object):
 5     """description of class"""
 6
 7     tilesAmount = 0
 8     frames = 0
 9     current = 0
10     sprite = 0
11     frameIndex = 0
12     loadedSpriteNr = 0
13
14     def Create(filePath,number = None):
15
16         if number != None: Sprite.loadedSpriteNr =
   number
17
18         try:
19             if os.path.isfile(filePath): sheet =
   pygame.image.load(filePath).convert()
20             else: sheet = pygame.image.load("notFound
   .gif").convert()
21
22             imgWidth, imgHeight = sheet.get_size()
23             spriteTiles = int(imgWidth / 300)
24             spriteTilesVert = int(imgHeight / 300)
25             cells = []
26             spriteOffset = 0
27
28             for r in range(spriteTilesVert):
29                 width, height = (300,300)
30                 for n in range(spriteTiles):
31                     rect = pygame.Rect(n * width, r
    * height,width,height)
32                     image = pygame.Surface(rect.size
   ).convert()
33                     image.blit(sheet, (0,0),rect)
34                     alpha = image.get_at((0,0))
35                     image.set_colorkey(alpha)
36                     if r == spriteTilesVert-1:
37                         if image.get_at((150,150
   )) != (0,0,0,255): cells.append(image)
38                         else: spriteOffset += 1
```

```
39                      else: cells.append(image)
40              playerImg = cells[0]
41              player = playerImg.get_rect()
42              player.center = (150,150)
43              spriteFrame = 0
44
45              Sprite.tilesAmount = spriteTiles *
   spriteTilesVert - spriteOffset
46              Sprite.frames = cells
47              Sprite.current = playerImg
48              Sprite.sprite = player
49          except:
50              pass
51
52          #return (spriteTiles * spriteTilesVert -
   spriteOffset,cells,playerImg,player)
53
54      def Cycle(frame,tilesAmt,cells):
55          spriteFrame = frame + 1
56          if spriteFrame >= tilesAmt: spriteFrame = 0
57          spriteImage = cells[spriteFrame]
58
59          Sprite.frameIndex = spriteFrame
60          Sprite.current = spriteImage
61
62          #return (spriteFrame,spriteImage)
63
64
65
```

```
 1 import pygame
 2 from pygame import gfxdraw
 3 from CText import Text
 4
 5 class UI(object):
 6     """description of class"""
 7
 8     def ProgressBar(surface,pos,width,height,color,
   text,minValue,maxValue,value):
 9         white = (255,255,255)
10         displayWidth = int(width/((maxValue-minValue+
   1)/(value-minValue+1)))-1
11
12         pygame.gfxdraw.aacircle(surface,pos[0],pos[1
   ],int(height/2),white)
13         pygame.gfxdraw.filled_circle(surface,pos[0],
   pos[1],int(height/2),white)
14         pygame.gfxdraw.aacircle(surface,pos[0]+width,
   pos[1],int(height/2),white)
15         pygame.gfxdraw.filled_circle(surface,pos[0]+
   width,pos[1],int(height/2),white)
16         pygame.gfxdraw.box(surface,(pos[0],pos[1]-int
   (height/2),width,height),white)
17
18         pygame.gfxdraw.aacircle(surface,pos[0],pos[1
   ],int(height/2)-2,color)
19         pygame.gfxdraw.filled_circle(surface,pos[0],
   pos[1],int(height/2)-2,color)
20         pygame.gfxdraw.aacircle(surface,pos[0]+
   displayWidth,pos[1],int(height/2)-2,color)
21         pygame.gfxdraw.filled_circle(surface,pos[0]+
   displayWidth,pos[1],int(height/2)-2,color)
22         pygame.gfxdraw.box(surface,(pos[0],pos[1]-int
   (height/2)+2,displayWidth,height-4),color)
23
24         Text.Write(surface,(pos[0],pos[1]-25),text,18
   ,"joy.otf",white)
25         Text.Write(surface,(pos[0]+width+30,pos[1]),
   str(value),18,"joy.otf",white,True)
26
27
```