```
"""
*********************************************************************************
 FILE:    figure.py

 AUTHOR:    Peter Campbell

 ASSIGNMENT:  Lab 3

 DATE:      10/01/12

 DESCRIPTION: This is a program with two top-level functions 'testFigure' and
 'motion'. Test figure takes two mouse clicks from the user and then draws
 a scaled figure into the rectangle bound by those click points. Another click
 makes the figure dissapear and a final click ends the program. Motion takes
 an input from the user that defines how many figures will be placed on the
 screen. The figures are then added in the same manner as testFigure, where
 the user clicks twice and the image is drawn in the rectangle bound by those
 points. Once all of the figures are added the program then promts the user to
 click the screen to set a destination point where the figures will all move
 towards.

*********************************************************************************
"""

from cs110graphics import *


def normalize(p1, p2):
    """This program takes two points that will be input by the user to define
  a rectangle and returns two new points for the upper left corner and lower
  right corner of that rectangle. This program is necessary for the scaling of
  the image so we are always basing it off of the upper left and lower right
  corners instead of just where the user clicks."""

    p1x = p1.getX()
    p1y = p1.getY()
    p2x = p2.getX()
    p2y = p2.getY()
    upper_x = min(p1x, p2x)
    upper_y = min(p1y, p2y)
    lower_x = max(p1x, p2x)
    lower_y = max(p1y, p2y)
    ul = Point(upper_x, upper_y)
    lr = Point(lower_x, lower_y)
    return (ul, lr)

def makeFigure(p1, p2):
    """This program takes two points that have been 'normalized' to define a
  rectangle and scales an image inside that rectangle. All of the scaling was
  based off of an initial figure drawn in a 500 by 500 window."""

    #calculate the x-length and y-length of rectangle bound by points p1 and p2.
    xlng = p2.getX() - p1.getX()
    ylng = p2.getY() - p1.getY()

    #scale the graphical elements of an image created in a 500 by 500 window so
    #that the entire image is scaled into the rectangle defined by the user's
    #click's
    area1 = Polygon(Point(p1.getX(), 275. / 500. * ylng + p1.getY()),
                    Point(175. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                          p1.getY()),
                    Point(175. / 500. * xlng + p1.getX(), 1 * ylng +
                          p1.getY()),
```

Good

```
                        Point(p1.getX(), p2.getY()))
    area2 = Polygon(Point(175. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                          p1.getY()),
                    Point(p2.getX(), 350. / 500. * ylng + p1.getY()),
                    Point(p2.getX(), p2.getY()),
                    Point(175. / 500. * xlng + p1.getX(), p2.getY()))
    area3 = Polygon(Point(200. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                          p1.getY()),
                    Point(p2.getX(), 300. / 500. * ylng + p1.getY()),
                    Point(p2.getX(), p2.getY()),
                    Point(200. / 500. * xlng + p1.getX(), p2.getY()))
    sky = Rectangle(xlng, ylng, Point(xlng / 2. + p1.getX(), ylng / 2. +
                                      p1.getY()))
    sun = Circle(min(xlng, ylng) * 40. / 500.,
                 Point(275. / 500. * xlng + p1.getX(), 75. / 500. * ylng +
                       p1.getY()))
    mtn = Polygon(Point(p1.getX(), 275. / 500. * ylng + p1.getY()),
                  Point(25. / 500. * xlng + p1.getX(), 200. / 500. * ylng +
                        p1.getY()),
                  Point(50. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                        p1.getY()))
    mtn2 = Polygon(Point(35. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                         p1.getY()),
                   Point(85. / 500. * xlng + p1.getX(), 150. / 500. * ylng +
                         p1.getY()),
                   Point(135. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                         p1.getY()))
    mtn3 = Polygon(Point(100. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                         p1.getY()),
                   Point(130. / 500. * xlng + p1.getX(), 205. / 500. * ylng +
                         p1.getY()),
                   Point(160. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                         p1.getY()))
    mtn4 = Polygon(Point(130. / 500. * xlng + p1.getX(), 275. / 500. * ylng +
                         p1.getY()),
                   Point(190. / 500. * xlng + p1.getX(), 100. / 500. * ylng +
                         p1.getY()),
                   Point(250. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()))
    mtn5 = Polygon(Point(200. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()),
                   Point(250. / 500. * xlng + p1.getX(), 175. / 500. * ylng +
                         p1.getY()),
                   Point(300. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()))
    mtn6 = Polygon(Point(270. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()),
                   Point(310. / 500. * xlng + p1.getX(), 160. / 500. * ylng +
                         p1.getY()),
                   Point(350. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()))
    mtn7 = Polygon(Point(290. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()),
                   Point(370. / 500. * xlng + p1.getX(), 75. / 500. * ylng +
                         p1.getY()),
                   Point(450. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()))
    mtn8 = Polygon(Point(380. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()),
                   Point(440. / 500. * xlng + p1.getX(), 175. / 500. * ylng +
                         p1.getY()),
                   Point(500. / 500. * xlng + p1.getX(), 300. / 500. * ylng +
                         p1.getY()))
```

```
127     road = Polygon(Point(25. / 500. * xlng + p1.getX(), p2.getY()),
128                    Point(350. / 500. * xlng + p1.getX(), 315. / 500. * ylng +
129                          p1.getY()),
130                    Point(355. / 500. * xlng + p1.getX(), 317. / 500. * ylng +
131                          p1.getY()),
132                    Point(200. / 500. * xlng + p1.getX(), p2.getY()))
133
134     #give the graphical elements color and if necessarry depth or border color.
135     area1.setFillColor("darkgreen")
136     area1.setDepth(20)
137     area1.setBorderColor("darkgreen")
138     area2.setFillColor("darkgreen")
139     area2.setDepth(20)
140     area3.setFillColor("darkgreen")
141     area3.setDepth(20)
142     sky.setFillColor("lightblue")
143     sun.setFillColor("orange")
144     sun.setDepth(0)
145     sun.setBorderColor("orange")
146     mtn.setFillColor("grey")
147     mtn.setDepth(40)
148     mtn2.setFillColor("grey")
149     mtn2.setDepth(35)
150     mtn3.setFillColor("grey")
151     mtn3.setDepth(45)
152     mtn4.setFillColor("grey")
153     mtn4.setDepth(49)
154     mtn5.setFillColor("grey")
155     mtn5.setDepth(45)
156     mtn6.setFillColor("grey")
157     mtn6.setDepth(35)
158     mtn7.setFillColor("grey")
159     mtn7.setDepth(40)
160     mtn8.setFillColor("grey")
161     mtn8.setDepth(30)
162     road.setFillColor("black")
163     road.setDepth(10)
164
165     #calculate the x and y coordinates of the center point of the scaled image.
166     cx = (p1.getX() + p2.getX()) / 2.
167     cy = (p1.getY() + p2.getY()) / 2.
168
169     #create a list of the center point followed by all of the graphical elements
170     #and then return that lsit.
171     myFig = [Point(cx, cy), area1, area2, area3, sky, sun, mtn, mtn2, mtn3, mtn4
172              , mtn5, mtn6, mtn7, mtn8, road]
173     return myFig
174
175 def addFigure(figure, win):
176     """This program takes a figure that has been scaled properly to fit inside a
177   user defined rectangle and adds it to a window."""
178
179     #select only the graphical elements of the list created by 'makeFigure' and
180     #add them to a window.
181     for i in range(1, len(figure)):
182         win.add(figure[i])
183
184 def removeFigure(figure, win):
185     """This program takes a figure that has been added to a window and removes
186   it from that window."""
187
188     #select all of the graphical elements from the list created by 'makeFigure'
189     #that have previously been added to a window and remove them from that
```

Nice image

```
190         #window.
191         for i in range(1, len(figure)):
192             win.remove(figure[i])
193
194     def testFigure():
195         """This is a program that takes the input of two points from the user and
196       uses sub-programs to normalize those points, make a scaled figure, add that
197       figure to a winow, and then remove it from the window."""
198
199         #make a window for the figure to be added to
200         win = Window(800, 800)
201
202         #get two points from user clicks that will define the rectangle that the
203         #figure is scaled into.
204         p1 = win.wait("Click the mouse to define the first boundary corner" +
205                       " for the figure").getMouseLocation()
206         p2 = win.wait("Click the mouse to define the second boundary corner" +
207                       " for the figure").getMouseLocation()
208
209         #apply sub-programs to make the scaled figure, add it to the window, accept
210         #a click to remove it from the window, and then accept a click to close the
211         #window.
212         p1, p2 = normalize(p1, p2)
213         fig = makeFigure(p1, p2)
214         addFigure(fig, win)
215         win.wait("Click the window to remove the figure")
216         removeFigure(fig, win)
217         win.wait("Click the window to close the window")
218         win.close()
219
220     def moveFigure(figure, dx, dy):
221         cx = figure[0].getX()
222         cy = figure[0].getY()
223         for item in figure[1:]:
224             item.move(dx, dy)
225             update()
226             figure[0] = Point(cx + dx, cy + dy)
227
228
229
230     def motion():
231         """This is a program that takes an input from the user which defines how
232       many figures will be drawn in the window. It uses sub-programs to normalize
233       points that define the rectangles that the figures will be drawn in, make
234       the scaled figure, and then add it to the window. After the figures have
235       been added the program takes a destination point from a user click and then
236       moves the added figures towards that point."""
237
238         #create a window, ask the user how many figures they want to create, and
239         #create an empty list for the lists of the figures.
240         win = Window(800, 800)
241         numberFigures = int(raw_input("How many figures do you want to make?"))
242         figList = []
243
244         #start a loop to add the number of figures that the user inputed and add
245         #each list of figures to figList
246         for _ in range(numberFigures):
247             p1 = win.wait("Click the mouse to define the first boundary corner" +
248                           " for figure " + str(_ + 1)).getMouseLocation()
249             p2 = win.wait("Click the mouse to define the second boundary corner" +
250                           " for figure " + str(_ + 1)).getMouseLocation()
251             p1, p2 = normalize(p1, p2)
252             fig = makeFigure(p1, p2)
```

do this in motion... all move together

could be more Good!

```
253             figList = figList + [fig]
254             addFigure(fig, win)
255
256         #recieve a mouse click to set the destination of the figures
257         destination = win.wait("Click the mouse to set the" +
258                                " destination").getMouseLocation()
259
260         #start a loop to move the each graphical element of each figure that has
261         #been added to the screen. This loop also changes the center point of each
262         #figure so that the distance that it moves changes each time that it goes
263         #through the loop.
264
265         for _ in range(40):
266             for figure in figList:
267                 dx = (destination.getX() - figure[0].getX()) / 16.
268                 dy = (destination.getY() - figure[0].getY()) / 16.
269                 moveFigure(figure, dx, dy)
270         update() ← would make less jerky (or not at all)*
271         #recieve a click to close the window
272         win.wait("Click the mouse to close the window")
273         win.close()
274
275 if __name__ == "__main__":
276     StartGraphicsSystem(motion)
277
278
279
280
```