Algorithms for Game Design Homework #2

Completely Optional! Will not be collected or graded.

---

1) Fonts

Write a simple word wrap function and place it in a Pygame window.

```
def word_wrap(rect, font, text, color):
```

Let's assume the `text` is English language (i.e. words break on spaces) and we want the words to be rendered, starting in the top-left of the `rect`. Use the given `font` object to determine the size of each word and place as many as you can on each line. Be careful to properly handle newline characters.

Only include words that entirely fit within the `rect`.

Return a surface, with the same size as the `rect`, with all of the text rendered in it.

I've given you some starter code in `hw_wordwrap.py` for you.

For maximum coolness points, detect keyboard presses of numbers 1-5 and re-render the text with a different font for each key press.

Or, define the rect based on mouse press and mouse release, so the user can control the size of the rect.

---

2) Collision Detection

Play around with the various versions of `sprite.py` to understand how sprites work. Take notes and turn in a couple of sentences about what you learned.

In `sprite1.py`: How does the sprite get drawn, centered at the place the mouse was pressed?

In `sprite2.py`: Will I eventually run out of memory if I keep clicking the mouse button?

In `sprite3.py`: This appears to do the same thing as `sprite2.py`. Does it do it more efficiently? Make sure you understand how!

In `sprite4.py`: Why does Mario pass behind the coins (instead of in front of them) when he falls through the same space? Can you change the code so he stays in front?

In `sprite5.py` and `sprite6.py`, try each of the different collision detection callback functions (i.e. `collide_*`). Set the attributes in each sprite so the collision is as good as you can get it.

---

3) Textbook

Make sure you have read chapters 0-3 of the textbook.