CMSC 143: Object-Oriented Programming with Robots

Lab 9: Film Strip Fun

Due November 3, 2016

This lab reviews some methods for manipulating film strips — Python lists of Pictures. Create and test the following functions:

1. `saveMovie(film, base)`: saves a series of frames in the list `film` as jpg files using `savePicture`. The files should be named `base-#.jpg` where the number increases starting at 0; `base` could be any string.

   `saveMovie([p1, p2, p3], 'film')` creates files named `film-0.jpg`, `film-1.jpg`, `film-2.jpg`.
2. `loadMovie(base, number)`: loads a set of consecutive images from files named `base-#.jpg` and returns them as a filmstrip (a list).

   `loadMovie('film', 3)` would return `[p1, p2, p3]` assuming previous `saveMovie` was used.
3. `repeatScene(scene, n)`: returns a new filmstrip with the list `scene` repeated `n` times.
4. `repeatFrame(frame, n)`: returns a filmstrip with the single Picture, `frame`, repeated `n` times.
5. `splice(scene1, scene2)`: returns a new filmstrip that combines two scenes back to back.
6. `append(film, scene)`: modifies `film` by tacking on `scene` to the end of the `film`.
7. `prepend(film, scene)`: modifies `film` by tacking on `scene` to the beginning of the `film`.
8. `reverse(film)`: Returns a new list with the frames reversed. For example, `reverse([p1, p2, p3])` would return the list `[p3, p2, p1]`.
9. `playSubtitles(film, subtitles)`: narrates the filmstrip using `show` and `speak`. These lists should be of equal length. `speak('Hello', False)` will block until the text has been fully read.

All the functions should use `assert` to identify and assure pre- and post-conditions. A file named `lab9.py` is available on Moodle with some tests; you should fully document the current tests and add more.

```
assert CONDITION, DESCRIPTION_OF_WHAT_WENT_WRONG
```

Learning Objectives

- Manipulate lists
- Practice using assertions

Deliverables

Submit an electronic copy of your lab using moodle and bring a hardcopy to class. Your program should have your name, email, assignment description, the date, and collaboration statement at the top of the file as a comment. Your submission should be a zip file that expands to a folder with one file:

```
cmsc143-lab9-LASTNAME-FIRSTNAME
      lab9.py
```

```
from Myro import *
from Graphics import Picture, Pixel

def takeMovie(n):
    '''return a list of n pictures'''
    assert n > 0, "number of frames should be greater than 0"
    lst = []
    for i in range(n-1):
        lst.append(takePicture())
    return lst

def picsEqual(p1, p2):
    '''return true if p1 and p2 are the same Picture otherwise return false'''
    assert type(p1) == Picture, "p1 is not a Picture"
    assert type(p2) == Picture, "p2 is not a Picture"

    if getWidth(p1) != getWidth(p2) or getHeight(p1) != getHeight(p2):
        return False

    for px1 in getPixels(p1):
        x = getX(px1)
        y = getY(px1)
        px2 = getPixel(p2, x, y)
        if abs(getGray(px1) - getGray(px2)) > 50:
            return False

    return True

if __name__ == "__main__":

    movie = takeMovie(15)
    assert len(movie) == 15, "takeMovie not returning right number of frames"

    saveMovie(movie, "masterpiece")
    lmovie = loadMovie("masterpiece", 15)
    assert len(movie) == len(lmovie), "mismatch between saved and loaded movies"
    assert picsEqual(movie[0], lmovie[0])

    movie3 = repeatScene(movie,3)

    assert len(movie3) == 45
    assert picsEqual(movie3[15], movie[0]) and picsEqual(movie3[30], movie[0])
    assert len(movie) == 15
    assert len(repeatFrame(movie[0], 10)) == 10
    assert len(splice(movie, movie)) == 30
    assert len(movie) == 15
    append(movie, movie)
    assert len(movie) == 30
    assert picsEqual(movie[0], movie[15])
    rmovie = reverse(movie)
    assert picsEqual(movie[0], rmovie[-1])
```