

CSCI 136 Assignment – Ultima 0

February 24th, 2023

Due Date: Friday, 3/3/2023 11:59PM

Points: 30

Topic: Recursion, Game Development, Prerequisites for Threading Lab

In this assignment, you will be creating the starting point for a tile-based role-playing game similar to the famous Ultima series ([https://en.wikipedia.org/wiki/Ultima_\(series\)](https://en.wikipedia.org/wiki/Ultima_(series))). You will create a recursive algorithm to handle a variable power torch that lights your Avatar's surroundings. In the subsequent lab you will add threaded monsters to your game, so build carefully. You need to add all the methods specified in this assignment.

After following all instructions below, please submit your Python scripts (`Avatar.py`, `Tile.py`, and `World.py`) in Moodle. Upload these as individual files – do not zip them. Be sure the script files have header sections that include the file name, your name, credits, a description of the file, the class and other useful information. Make sure your classes and all methods are thoroughly commented.

- `Avatar.py`
- `Tile.py`
- `World.py`

Grading:

This assignment is worth 30 points. You will be graded according to the following criteria:

Grade Item	Points
Program Compiles and Runs	4
Comments on All Classes and Methods	4
Tile	6
Avatar	8
World	5
Used Recursion for Torch Light	3
Total	30

Basics

The game is played on a rectangular grid of tiles. The grid of tiles is specified in a text file. Your Avatar moves around by moving in one of the cardinal directions (north, south, east or west). The Avatar is not allowed to move through certain features (water or mountains). It is nighttime, and your Avatar has a variable brightness torch. The torch cannot see through certain features such as forests.








Files (ultima.zip)

The file `ultima.zip` contains the set of graphics tiles you will use in this assignment. It also contains stub/starter versions of the `Tile.py`, `World.py`, `Avatar.py` scripts. You are also given a fully functional version of the main game program `Ultima.py`.

Tile.py

A `Tile` object represents an individual position in the Ultima world. All tiles are the same size: 16 by 16 pixels. Tiles know things like what type of tile it is and whether it is currently lit by the torch. A tile can do things like tell whether it is lit, set whether it is lit or not, draw itself, determine if the Avatar can walk through it, and determine if light passes through it. Tiles should default to not being lit.

Here are the details of the types of tiles your program needs to support:

Name	Filename	Image	Opaque?	Passable?	String code
Brick floor	brickfloor.gif		No	Yes	B
Lava	lava.gif		No	Yes	L
Water	water.gif		No	No	W
Forest	forest.gif		Yes	Yes	F
Grasslands	grasslands.gif		No	Yes	G
Mountains	mountains.gif		Yes	No	M
Stone wall	stonewall.gif		Yes	No	S

If a tile is not lit, you should draw it using the supplied `blank.gif` image.

Here is the specification you should implement for the `Tile` class:

```
class Tile
-----
# Create a new tile based on a String code
__init__(self, string code)

# Return whether this Tile is lit or not
boolean getLit(self)

# Change the lit status of this Tile
setLit(self, boolean value)

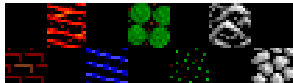
# Draw at index position (x, y)
draw(self, int x, int y)

# Does this type of Tile block light?
boolean isOpaque(self)

# Can the Avatar walk on this Tile?
boolean isPassable(self)
```

You are provided with a test main function for `Tile`. Here is the output:

```
% python Tile.py
0 0 : lit True opaque False    passable True
0 1 : lit False opaque False   passable True
1 0 : lit False opaque False   passable True
1 1 : lit True opaque False    passable True
2 0 : lit True opaque False    passable False
2 1 : lit False opaque False   passable False
3 0 : lit False opaque True    passable True
3 1 : lit True opaque True     passable True
4 0 : lit True opaque False    passable True
4 1 : lit False opaque False   passable True
5 0 : lit False opaque True    passable False
5 1 : lit True opaque True     passable False
6 0 : lit True opaque True     passable False
6 1 : lit False opaque True    passable False
```



Avatar.py

An Avatar object represents a player in the game. An Avatar knows things like its current (x,y) position in the world and the current power of the Avatar's torch. The (x,y) position consists of indices: (0,0) is the lower-left tile, (1, 0) is one tile to east, (0, 1) is one tile to the north, etc. An Avatar can do things like get its (x,y) position, change its location, get its current torch power, increase/decrease its torch power, and draw itself. An Avatar's torch has a minimum torch radius of 2.0. The torch power changes in increments of 0.5. The torch starts with a default radius of 4.0.

Your Avatar data type must implement the following specification:

```
class Avatar
-----
# Create a new Avatar at index position (x,y)
__init__(self, int x, int y)

# Get the current x-position of the Avatar
int getX(self)

# Get the current y-position of the Avatar
int getY(self)

# Update the position of the Avatar to index (x,y)
setLocation(self, int x, int y)

# Get the current torch radius
float getTorchRadius(self)

# Increase torch radius by 0.5
increaseTorch(self)

# Decrease torch radius by 0.5, minimum is 2.0
decreaseTorch(self)

# Draw at the current position
draw(self)
```

You have been provided a test main function for Avatar. Here is the output:

```
% python Avatar.py
5 5 4.0
1 4 4.0
1 4 4.5
1 4 4.0
1 4 3.5
1 4 3.0
1 4 2.5
1 4 2.0
1 4 2.0
```

World.py

The World class represents all the tiles in the world as well as the Avatar. This class is responsible for handling all keystrokes from the main program in `Ultima.py`. The class knows things like all the `Tile` objects in the world and the `Avatar` object. The class can do things like handle keystrokes from the user, draw itself, and light a portion of the world. Your World data type must implement the following specification:

```
class World
-----
# Load the tiles and Avatar based on the given filename
__init__(self, String filename)

# Handle a keypress from the main game program
handleKey(self, char ch)

# Draw all the tiles and the Avatar
draw(self)

# Set the tiles that are lit based on an initial position (x,y)
# and a torch radius of r. Returns the number of tiles that were
# lit up by the algorithm.
int light(self, int x, int y, float r)

# Recursive lighting method called by light(). (x, y) is still
# the position of the avatar, and (currX, currY) is the
# position being considered for lighting.
int lightDFS(self, int x, int y, int currX, int currY, float r)
```

The constructor of World reads in a file using Python file I/O. Here is an example input file:

```
10 5
3 1
W W W W W G G G W W
W F W G W S G G G G
W F F G S L S G G G
W F F G G S G G G W
W W W F G G G G G G
```

The integers on the first line specify the width and height of the world. The integers on the second line specify the starting x- and y-index of the Avatar. The remaining letters give the code letters for each tile in the world.

The `handleKey()` method in your `World` class should handle the following keys:

w : Move the Avatar north.

Nothing should happen if the tile to the north is not passable or is off the map.

s : Move the Avatar south.

Nothing should happen if the tile to the south is not passable or is off the map.

a : Move the Avatar west.

Nothing should happen if the tile to the west is not passable or is off the map.

d : Move the Avatar east.

Nothing should happen if the tile to the east is not passable or is off the map.

+ : Increase the torch radius by 0.5, there is no maximum radius.

- : Decrease the torch radius by 0.5, subject to a minimum radius of 2.0.

The lighting algorithm is the crux of the assignment. You will need to implement a recursive helper method `lightDFS()` that gets called by the public `light` method:

```
lightDFS(self, int x, int y, int currentX, int currentY, float r)
```

What do you think DFS stands for in `lightDFS`?

The basic approach is to first set all the `Tile` objects to being unlit. Then start `lightDFS` at the Avatar's current position. The `lightDFS` method will call itself recursively for the positions to the north, south, east and west of the given position. Use these four directions only, do NOT recurse on the diagonals. Recursion allows the algorithm to spread across the map. The `lightDFS` method must retain the initial `(x, y)` starting position so it can calculate how far the `(currentX, currentY)` position is from it. You must be careful to limit the recursion with appropriate base cases:

Base case 1 – The current position is off the map.

Base case 2 – The current position has already been visited - the `Tile` is marked as lit.

Base case 3 – The current position is opaque.

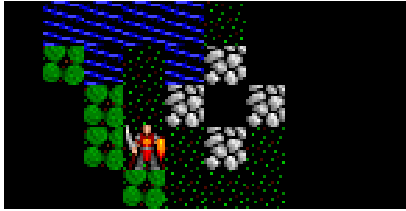
Base case 4 – The current position is outside the torch radius. A position is considered "outside" if the Euclidean distance between the `(x, y)` index of the Avatar and the `(x2, y2)` index of the tile is greater than or equal to the torch radius.

CSCI 136 Assignment – Recursive Graphics

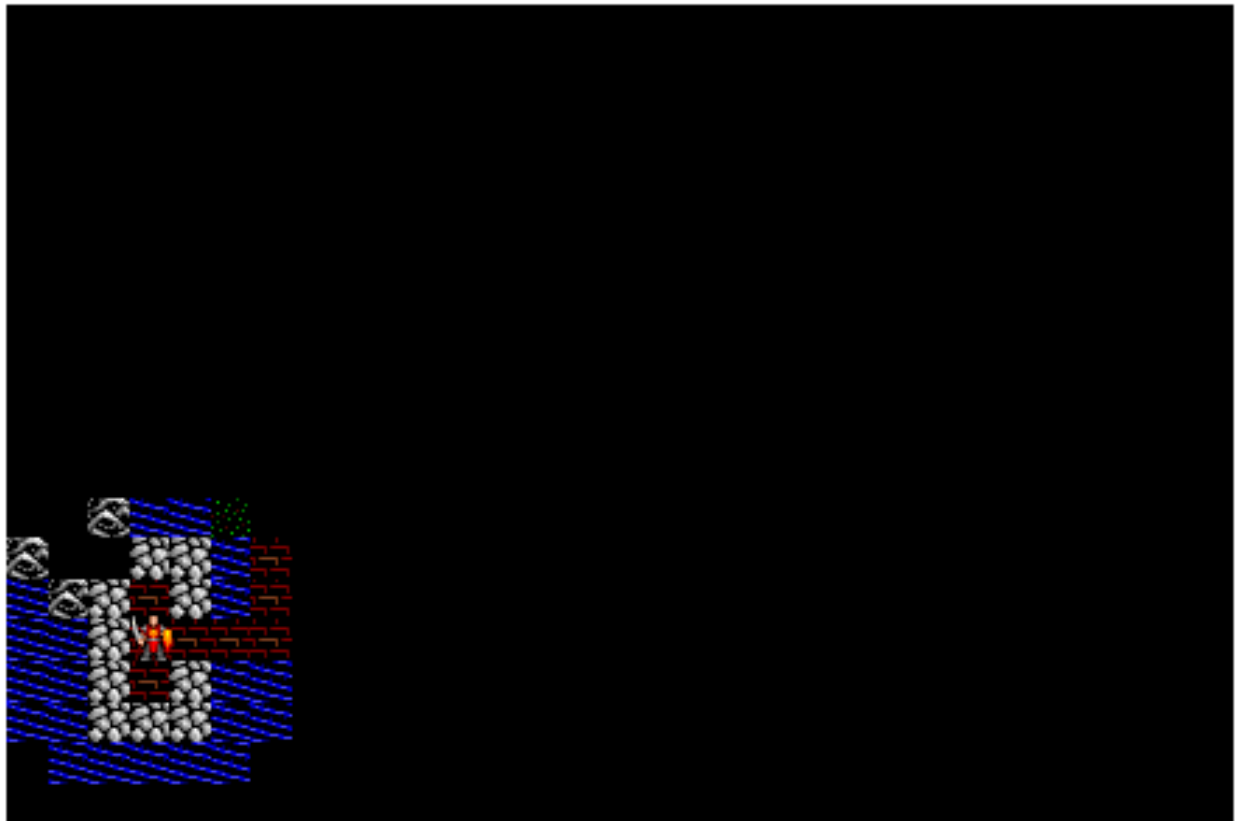
Opaque cells should be lit, but they should not propagate the search. This is what causes certain parts of the map to appear black despite being within the radius of the torch.

You are provided with a test `main()` function for `World.py`. Here are several runs: to show you the input parameters and result value.

```
% python World.py 10x5.txt  
light(3, 1, 4.0) = 23
```



```
% python World.py 30x20.txt  
light(3, 4, 4.0) = 42
```



Helpful Hints:

Do I need to follow the given specifications? Yes. You must implement all methods as described. You may add additional methods if they are helpful to your code.

I use the number 16 in multiple spots in my code. Is that okay? You should be able to declare the tile size in the program. If you do it this way, it will make it easy if in the future you need to change your program to use a different tile size.

My tiles are all offset and appear in the borders of the drawing window. The `StdDraw.picture()` method expects the center (x,y) position of the picture, not the lower-left corner. Adjust accordingly to get things to line up.

The blank areas of my map have strange lines. What is going on? This is caused by drawing black squares or rectangles using `StdDraw`. Draw a blank tile using the provided `blank.gif` file instead.

Do I need to modify the size of the drawing window? Yes. Each tile measures 16 pixels by 16 pixels, so you need to size the drawing window to accommodate this by calling `StdDraw.setCanvasSize()`. You should do this one time, so an appropriate place to do it would be in your `World` constructor since it is only called once in `Ultima`.

Do I need to modify the `StdDraw` coordinate system? Not necessarily, but you can and it may make things easier. As with the window size, you should do this in the constructor of `World`.

Can I send pixel coordinates as the x and y parameters to methods in `World`, `Tile`, and `Avatar`? No. The specification indicates that these are the integer index positions of the game tile, not pixels or other types of coordinates.

Submit Your Work

After following all instructions above, please submit your Python scripts (`Avatar.py`, `Tile.py`, and `World.py`) in Moodle. Upload these as individual files – do not zip them. Be sure the script files have header sections that include the file name, your name, credits, a description of the file, the class and other useful information. Make sure your classes and all methods are thoroughly commented.

- `Avatar.py`
- `Tile.py`
- `World.py`

EXTENSIONS

You will not submit the answers to these extensions as part of your assignment submission. You should work through and understand them to learn more and practice. You may be held responsible for the content within, so you really should work through them. With that said, this are outside the scope of what will be submitted for this lab. Do not change your original submittal to address these extensions.

There are no extensions for this lab. Do your best to get it right as it will provide a basis for the next lab. If you want to experiment, do it separate from what you submit.