Minestomper

Our neighbour country has declared a war agaist us and laid a minefield along the current border. We nee train some fools... *ahem*, pioneers to stomp... err, sweep the field so our forces can advance freely. And because field training is so 90s we want to be able to train our pioneers with a high tech minesweeper simulator. Your commanding officer has told us that you know Python so your task is to make that simulat for us. Your country needs you soldier, are you ready?

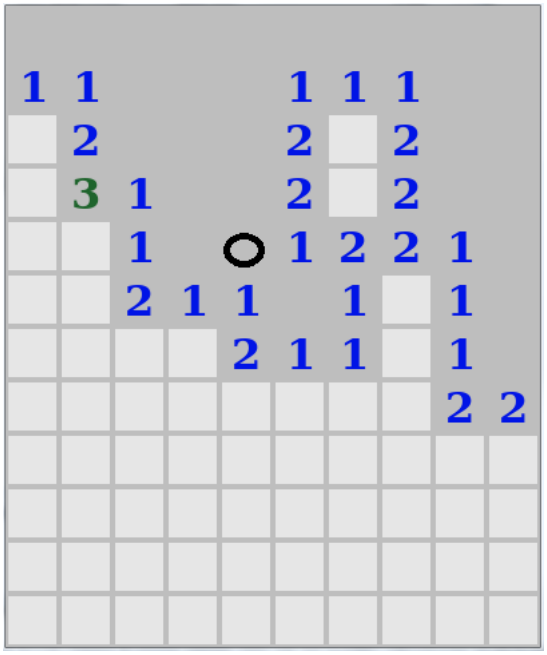
Game Setup

The minefield is a two dimensional rectangle shaped grid that contains mines. The player must be able to determine the dimensions of the field and the amount of mines on the field. All tiles on the field are initial unrevealed and the mines are on randomly selected tiles. Other tile types include a number tile, which displays the number of mines surrounding it, and an empty tile.

Gameplay

The player can choose one tile at once by clicking it with the mouse. The chosen tile is opened and its con is shown to the player. Then one of the following happens:

- If the opened tile contains a mine the game is over and the player lost.
- If one or more of the tiles next to (i.e. horizontally, vertically or diagonally) the opened tile contains a n the number of the mines in the surrounding tiles is shown in the opened tile.
- If there aren't any mines in any of the tiles next to the opened tile (i.e. it is an empty tile), all the surrounding tiles are opened and all the tiles next to those tiles are opened and so on until the border the minefield is reached or the opened tile is next to a mine (i.e. is a number tile). The tiles next to min are also opened but not the tiles beyond them, regardless of their contents.



Example of opening tiles. Circle marks the clicked spot

Ending

The game can end two different ways:

- If the opened tile contains a mine the player lost.
- If all mineless tiles on the minefield are opened, the player won.

You might want to do a little more research on the said game before you rush into making one yourself. Fo example: playing it for yourself certainly wouldn't hurt, Google could help you with that. The following instructions may also contain some important information regarding how you can pass this course, so kee reading.

Acceptable Implementation

In order for your project to be accepted, it needs to show the understanding of the couse contents. Below checklist you can use to determine whether you've fulfilled the requirements.

- The game correctly implements the minesweeper rules described above.
- The game works with user-selected dimensions - make sure to test rectangles that are not squares (i.e should work even when length and width are not equal)
- The game situation is shown to the player with a clear graphical representation
- The player can make moves with the mouse; the canonicality of the moves is checked and illegal move are prevented
- The game ends when it should (review the "Ending" section)
- The game keeps a record of the played games: when the game was played (the date and time), the duration (in minutes and turns) and the outcome (was the game won or lost and how many mines ther were left on the field).
- The game contains a main menu where one can choose to play a new game, quit or look at the statisti note! the menu can be text-based in the terminal window, only the game window itself needs to use graphics.

Graphics

We've created a small helper library for creating graphics in a game window with more or less the same m as everything else we've done in the course - by calling functions. You don't have to use this library but th game must have graphics, implemented one way or another - the reign of terminal minesweepers is over! library module has rather extensive docstrings that should provide assistance about its use. Some additio instructions can be found below.

Deployment

sweeperlib.py

sprites.zip

Download the library and graphics zip from above. The images should be placed into a folder called sprite inside your project folder - it's a zipped folder so don't create another sprites folder when extracting it. Yo need to also install Pyglet because our small library is built on top of it. Pyglet is a very neat library in gene for making 2D games with Python. If you want, you can also download the script that was used to generat the graphics in the zip (requires PyCairo).

Game Libraries 101

Whereas programs so far have had their own main loop (usually with `while True`), game libraries typically hide the main loop under the hood, and it is usually much more complex. It's been hidden so that you don' need to worry about it. When using a library, the game is usually implemented through handler functions t the main loop calls when certain triggers happen. For instance, mouse clicks can have a handler attached - handler function will then be called whenever the user clicks a mouse button. Another very common hand is a drawing handler to draw the game's graphics, which is called whenever the game screen needs to be updated.

You will not be creating a main loop in your own program. Instead, you will be implementing handler functions. These handlers must be attached to events. This can be done with functions in our library (nam like: set_some_handler). The docstring for each of these describes what kind of a handler function is requi The general workflow of using the library is:

1. load game graphics
2. create game window
3. define and register handler functions
4. start the game

After this the game will run in the handler functions. There's a very simple example of drawing graphics at end of the library module. The main program is a small piece of code that draws all sprites in some order o one row inside the window.

The most important fact that needs to be accounted for is that handlers are called by the main loop runnir out of your reach. This means you cannot cotrol what arguments your functions are given. As a result, the game's state (like the field) must be smuggled to the handlers in some other way. The preferred way to do is to create a main program level dictionary for the state. A dictionary defined in the main program can be accessed and modified in any and all functions of the program due to its mutable nature.

Return Box

Deadline: 2024-08-31 23:59

119.00 / 152

1 answer
Frequently Asked Questions
Assessment Criteria

You can return your project here. If you're returning multiple files, you can either zip them or use `Ctrl` wh selecting files to upload many at once.

Jos pakkaat tiedostoja, pakkaa ne zip-muodossa!
Allowed filenames: *.py, *.zip
Submit your files here: Choose Files No file chosen
Send answer You have already answered this task correctly.