Jeff’s Lair - User Manual

Document 1

Introduction

Welcome to Jeff’s Lair.

You have been captured by the evil wizard Jeff, turned into a crocodile and put in a dungeon.

You must escape fast and avoid the rats.

Game play

Click in the name box and enter your name. Click on “Start” to start the game.

Use left, right, up and down arrows to steer your crocodile around the maze.
But you will need the keys. Collect all 4 to unlock the door.

See how fast you can get to the exit to escape (it’s in the bottom right) but watch out for the rats.
If they touch you, you lose.

You only have 100 seconds, so go as fast as you can!

Press escape to exit at any time - don’t worry your score will be kept!

Scoring

See how fast you got out of the maze with all the keys avoiding the rats.
Press any key to try again to beat your best score.
Can you make it to the leader board?

Jeff's Lair - Installation Guide

Document 2

Tested on:

- Apple Mac
- Mojave
- Python 3.7
- PIP (should be installed with Python - please upgrade to the latest version)
- Pygame (Type in Terminal: python3 -m pip install -U pygame --user)

To run the game:

1. Dowload all files in github repository from .tar file
2. Run "run_jeff.py"

Running slow on your Mac?

Because of the resolution of the display, Jeff's Lair may run slower on an Apple Mac.
You can try this to speed it up:

1. Run the program
2. In the dock you will see a snake with controllers in his mouth. Right click him.
3. Go to Options and click "Show in Finder"
4. Finder will open and you will see a the python application.(Mine was in the shape of rocket with the idle symbol on it.)
5. Right click the python application and click "Get Info".
6. Check the box "Open in Low Resolution"

Jeff’s Lair - Maintenance Guide

Document 3

Jeff’s lair is a dungeon maze game built using the pygame module https://www.pygame.org

The main game initialisation loop and setup constants are in run_jeff.py.
The setup of the maze is text file driven (0=corridor, 1=wall, 2=additional wall block locations)
The main character and monsters movements and collision detection are in Hero.py
All resources - images,maps and sounds are in the resources folder.

Module: run_jeff.py

This is code that runs the game.

Constants

All global constants held here to fix resource locations and gameplay defaults.

Class: Game

Main game object built here

.__init__

Sets up the game with given parameters:
• Screen size: 800x600 (global constants)
• How big the tiles on the maze are in px: 32px (global constants)
• Title on maze window: Jeff’s Lair (global constants)
• Where the character will be placed at start: 32,32px (global constants)

Runs __init_pygame
Runs __init_game
runs .update

.__init_pygame

Loads pygame module
• Sets the timer

.__init_game

• Initialises the map
• Initialised hero and rat

.collide

Works out if 2 rectangles have hit each other, returns true if they do.

.update

This is the main game loop. There are 2 possible ways to finish:

game_over== True

Player finished maze, display win image and show score and nickname. Hold until quit/escape.

lose_game == True

Player caught by rat, display lose image. Hold until quit/escape.

• Make the clock update every second
• Make the clock tick
• detect collisions

Module: Hero.py

Class: Hero

Can be either hero or monster (rat)

.hero_moving

Moves the character left right up and down checking for screen edges, walls and winning square.

Checks for touching the rat and returns True if caught.

.monster_move

Moves the monster towards the character, returns True if caught.

Module: GameMap.py

Class: Block

A rectangle to form the list of rectangles that represent maze walls.

Class: GameMap

Creates the logical game map from text file

.draw_map

Put map on screen.

.make_block_group

Returns a list of rectangles that represent maze walls.

Module: Startbox.py

Function: Startbox

Creates a start screen for the user to enter their name and start the main game. Returns nickname.

Module: LeaderBoard.py

.Sort_Tuple:

Sorts the list of tuples (for the top 5 high scores)

Class LeaderBoard:

Creates the leaderboard object

.board_input_result

Reads top scores from Leaderboard.txt file. Adds latest score, orders list and removes the slowest. Then sames back to file.

Module: Key.py

Class Key:

Draws 4 keys