Jungle Game Version 22.11

Developer Manual

Group3

Yalu PAN

Shihao QI

Fengkai YU

Ming WANG

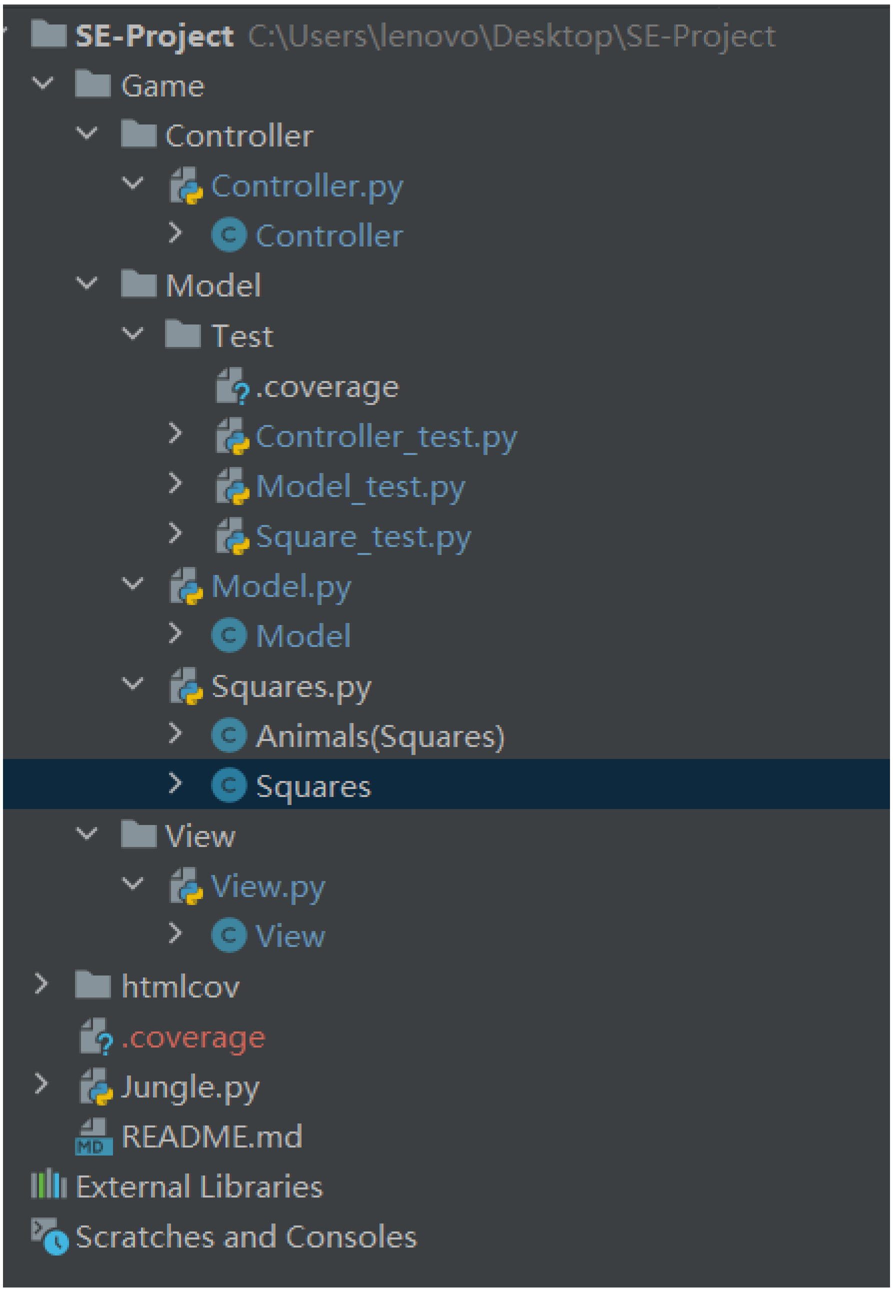
Introduction

This game system is a local standalone command-line Jungle Game supporting two players. This developer manual explains how to compile the code and build the game system.

The game is built with Python.

This developer manual is prepared for windows, using Python (version) to develop the game. Pycharm should be used to open our project.

Package Organization



```
SE-Project C:\Users\lenovo\Desktop\SE-Project
  Game
    Controller
      Controller.py
        Controller
    Model
      Test
        .coverage
        Controller_test.py
        Model_test.py
        Square_test.py
      Model.py
        Model
      Squares.py
        Animals(Squares)
        Squares
    View
      View.py
        View
  htmlcov
  .coverage
  Jungle.py
  README.md
External Libraries
Scratches and Consoles
```

Start the game

You can download the game system's source code from our github source code. https://github.com/YaluPAN/SE-Project.

To compile the project, input the command:

```
python Jungle.py
```

Then you run the file Jungle.py to start the game. Or you can just enter the jungle.py and start the game.

unittest Module

The unittest module provides a built-in testing framework to contract and run tests in python. It supports test automation, test aggregation, and independent tests.

The unit test suite of model Game/Model/Test contains Model_test.py and Square_test.py

We use this module as it is a required module to go through tests in python. We built a TestCase from unnitest.TestCase.

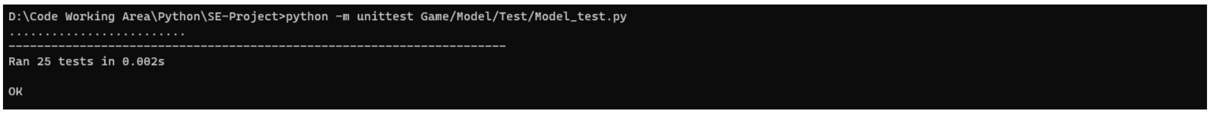
```
import unittest
class Model_Test(unittest.TestCase):
  ...
class Squares_Test(unittest.TestCase):
  ...
```

You can go to the Game folder and enter the Model folder and then enter the test folder and you can run the Model_test.py and Square_test.py for the model functions.

To run test coverage from command line

First you should download the Project zip and enter the correct root of this project.

Python -m unittest Game/Model/Test/Model_test.py

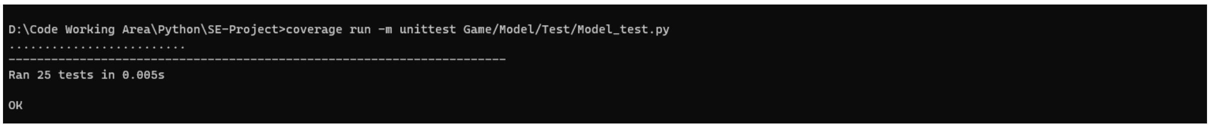


```
D:\Code Working Area\Python\SE-Project>python -m unittest Game/Model/Test/Model_test.py
.........................
----------------------------------------------------------------------
Ran 25 tests in 0.002s

OK
```

Then you run

```
coverage run -m unittest Game/Model/Test/Model_test.py
```



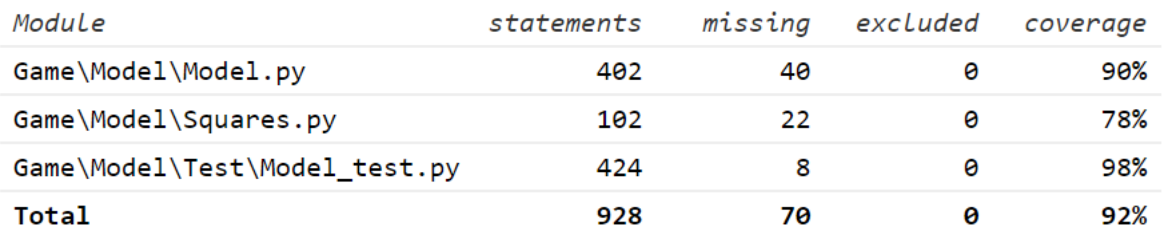
```
D:\Code Working Area\Python\SE-Project>coverage run -m unittest Game/Model/Test/Model_test.py
.........................
----------------------------------------------------------------------
Ran 25 tests in 0.005s

OK
```

If want to see the report results, you should input:

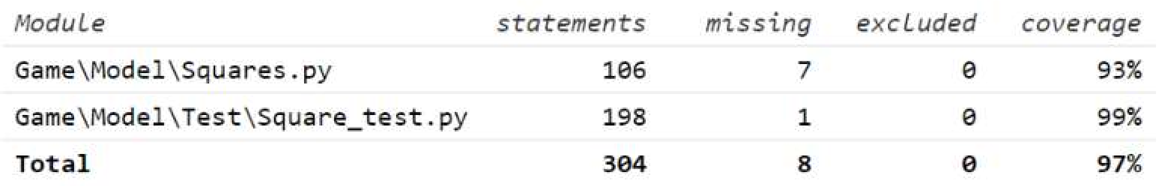
```
coverage report -m
```

This is the result for Model_test.py. Kindly remind that the coverage library will conclude all invoked files in the coverage calculation. Please only focus on the coverage of Model.py.



Module	statements	missing	excluded	coverage
Game\Model\Model.py	402	40	0	90%
Game\Model\Squares.py	102	22	0	78%
Game\Model\Test\Model_test.py	424	8	0	98%
Total	**928**	**70**	**0**	**92%**

And this is the result for square_test.py. Also Remind that this test coverage will contain all the p file which use python related operations



Module	statements	missing	excluded	coverage
Game\Model\Squares.py	106	7	0	93%
Game\Model\Test\Square_test.py	198	1	0	99%
Total	**304**	**8**	**0**	**97%**

If you want to see the report result in HTML format, you can then input

```
coverage html
```

Then it will appear in your project folder and you can have a better view to see the match situation