Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/30/20230071

The application of Python game algorithm in Rouge games

Zhewen Mi[1,3], Zunyi Zhang[2,4,5]

[1]Electrical and Electronic Engineering, Wenzhou University, Wenzhou, 325035, China
[2]Software Engineering, Wuhan Donghu University, Wuhan, 430000, China

[3]zhewenmi2001@163.com
[4]1317256735@qq.com
[5]corresponding author

Abstract. Our project's topic is algorithms for game design, which mainly describes a series of algorithms in the pygame library in Python, such as collisions, mazes, and a series of algorithms with different functions. The following is mainly about the ideas, processes, problems, and solutions encountered by our group's final project. In addition, we learned how to take part in the joint work and Python programming experience and the algorithms in pygame through this study and group cooperation.

Keywords: pygame, algorithm, python

1. Introduction

1.1. Game Design Background

With the rapid development of technology and economics, and the rapid growth of knowledge economies, entertainment has gradually become a communication between people and intelligent machines, and only-able-to-be-entertained games are becoming better and better in the network. There are numerous games, such as puzzle games, web games, fighting games, simulation games, etc. Recently, it has become increasingly popular to create games using Python programming and the Pygame library. To create a game of "Rouge" type, we have found an article [1] and a Chinese mobile game called "Spirit Knight," which is very in line with our ideas. We will use this as a reference to continue our game design [2,3].

1.2. Game Development Environment

Python development environment and Pygame library.

2. Game function

At the beginning of the game, players can choose existing characters and weapons and then enter the contest. During the game, players can move using the arrow keys and the mouse left button to shoot bullets. The mouse position on the screen controls the part of the shots, and there are also skills that are used at that position. The direction in which the skills are launched is towards the direction the player faces. The number and attributes of monsters in each level will increase, and a BOSS will appear after

DOI: 10.54254/2755-2721/30/20230071

five levels. Players must defeat all monsters in each scene to pass the class and clear all rooms to unlock the passage to the next level. When a monster beats a player, the game ends [4].

2.1. Game development process

2.1.1. Underlying code implementation. Since we were both new to Python game design, we didn't have much experience with the underlying code logic, which was a crucial part of the code. So, we searched relevant materials on the Internet to learn how to write the underlying code, referring to a "Tank Battle" code on the Internet [5]. We divided the initial code into ten classes: primary function, character, monster, player, bullet, skill, wall, explosion, music, and room [6,7].

2.1.2. Player class improvement. To realize the player character function in the game, we wrote the MyCharacter class. The central part of this class is to detect collisions between the player and enemies and whether the player has collided with the game room. The MyCharacter class code is well structured and uses two functions, myCharacter_moving, and myCharacter_hit_enemy, which are crucial to implementing the player character functionality. In the myCharacter_moving function, we used steps to move the player's image and rotate the weapon Angle based on the player's direction. In the myCharacter_hit_enemy role, we used a loop to iterate through the enemy list and used pygame.sprite.collide_mask to detect if there was a collision between the player and the enemy. This code was critical to our research because it gave us the foundation to implement the player character function, allowing us to operate effectively in the game [1].

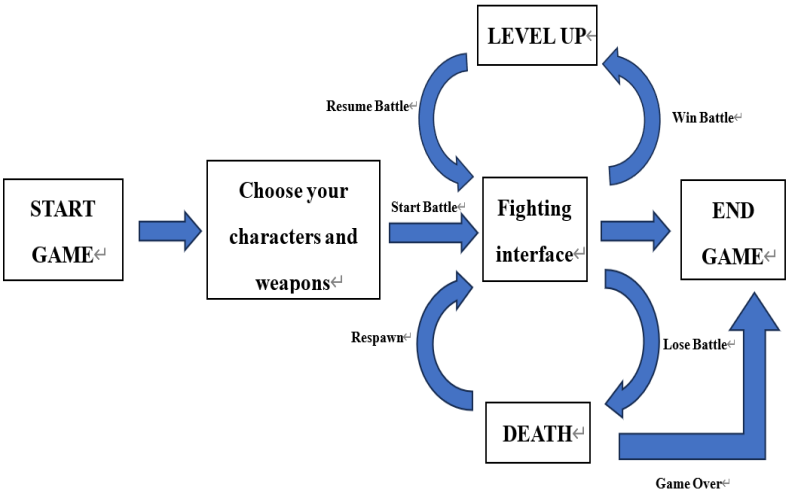

Figure 1. game progression

2.1.3. Monster class improvement. The class monster (Character) is a character subclass that represents the game's monsters. The constructor __init__() initializes the attributes of a monster object, such as its position (left and top), speed, health points (hp), and damage (dmg), as well as loads the necessary images for the monster's animation. The method randDirection() randomly generates the direction in which the beast will move, and the method randMove() moves the monster randomly. The method track () allows the beast to track and move toward the player's position. The method enemy_hit_myCharacter() checks for collision between the monster and the player character, while the method boom_hit_myCharacter() checks for collision between the monster and the player's bullets [1].

2.1.4. Skill class and bullet class implementation. The Skill class has attributes like image, direction, rect, speed, dmg, and live, which define the properties of a skill object. It also has methods to display the skill object, hit the enemy object, move, and hit the wall. The display skill () process is used to

DOI: 10.54254/2755-2721/30/20230071

display the skill object on the screen. The mySkill_hit_enemy() method detects a collision between the skill and enemy objects. The move () process moves the skill object across the screen. The hit-all () method sees a clash between the skill and wall objects.

The Bullet class has attributes like image, direction, rect, speed, dmg, angle, capacity, mask, and live, which define the properties of a bullet object. It also has methods to move off_screen, hitWall, displayBullet, and myBullet_hit_enemy. The move () process moves the bullet object across the screen. The off_screen() method detects if the bullet object has gone off the screen. The hit-all () method detects a collision between the bullet and wall objects. The displayBullet() method displays the bullet object on the net. The myBullet_hit_enemy() method sees a clash between the bullet and enemy objects.

2.1.5. The overall framework of the game is improved. The overall framework for running the game is four loops, the first loop, bool1, is the interface to start the game. When you click the Start button, we go to the second loop, while bool2 is the interface to select our players and weapons. Then we go to the third loop, while bool3, this is the primary cycle that the game runs through, going through monsters, bullets, etc., to keep the game running, and finally, while bool4, when you've killed all the monsters and come to the portal, go to bool4, select the attributes you want to enhance, and then go to the next level back to bool3, the primary cycle. We can also use this method to realize pause the whole game [8].

2.1.6. Add pictures and sound effects. The basics of the game are done, but the art and sound are part and parcel of a complete game. We have added different pictures for all characters, monsters, weapons, etc. The movement of characters and monsters is composed of three consecutive moving photos in the animation so that they can achieve the effect of energy when moving [9].

In implementing sound effects, we use the defined channel part of my game to ensure that a piece of audio will not be interrupted by other audio while playing. The set audio includes bullet firing, hits, character death, background music, etc.

2.1.7. Optimize details. Blood bars and other attributes are displayed on the screen. Characters and monsters of the various characteristics of the balance adjustment can ensure that the players play the game when the fairness and fun.

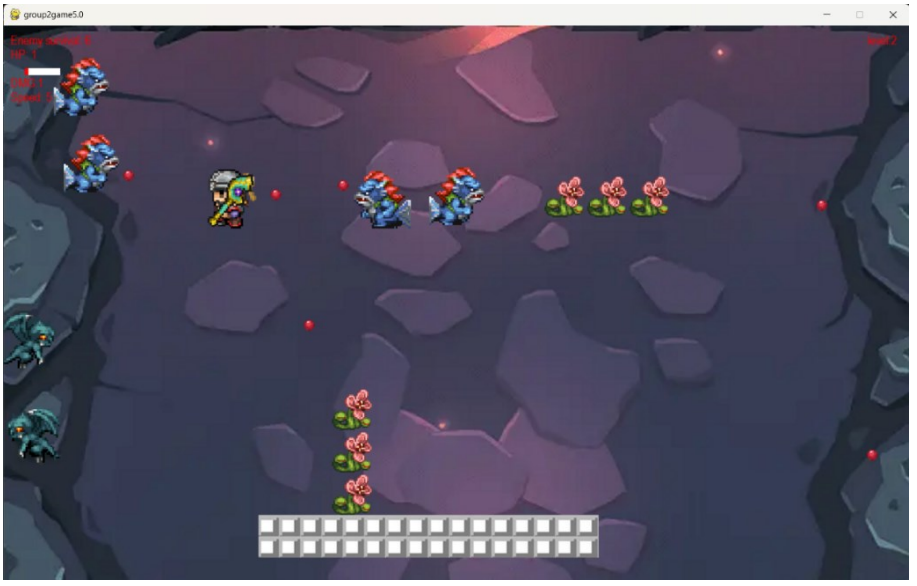

Figure 2. game visuals

DOI: 10.54254/2755-2721/30/20230071

2.1.8. Issue & Solution. Miles had his first problems when he was writing the underlying code. The program running, the contents of the primary function would be too large, so I separated a bit and created. For example, createEnemy only had the picture of the monster itself and added the monster to the list when creating. In blitEnemy, Miles added collision detection, survivability detection, etc., which made the program's writing more orderly and easier to change.

Another is the map design, we originally intended to use the maze algorithm in the article [10] to design the map, but later found that the map made the difficulty of the game reduced, so we simplified the map on this basis so that each level of the game is not too easy.

The next problem is how to set up different kinds of monsters. Both characters and monsters are sprites, making writing collision-related code easier. So, Miles used the monster class to add other monster movement features. For example, he had a monster that moved fast, and when it touched the player, the player would deduct health. So Miles defined a function in the Monster to judge whether the monster and the character collided, the player's health would be reduced, and the enemy would die. Then separate it with create, and a new monster appears. Zunyi and Miles wrote those monsters together, so the types of monsters are much more varied.

Zun Yi claims that he met many problems to keep the work running well. The first problem is the weapon's rotating and shooting to the mouse's position. Zun Yi utilized the function to draw a black line to show that the weapon's position was directed to the mouse. Therefore, if we moved the mouse, this line must have one end still in the Character's center, and another end shot bullet to the mouse position (the line circulates the character).

Second, because the initial code is regular, the character can only move on the X-axis or Y-axis, which made the bullet and can only move like a character. Hence, Zun Yi changed the regular movement of the nature and pellets to optimize the control handling to feel smoother.

The first problem Zun Yi met was making the line move like a circle, and one end of the line needed to be fixed in the character's center. He used the trigonometric function to represent the x and y coordinates of the endpoints of the line and the start points of the line's x and y coordinates equivalent to the character's x and y coordinates [11].

And then, we just needed to draw this line on the screen to make it visible and add the trigonometric function into the bullet move function to realize the bullet shooting from the line’s ending (the starting is the character’s center) to the mouse position.

After that, Zun Yi tried to make the monsters track the character's position because the professor pointed out that the anomalies should be more purposeful to move, like monitoring nature. So, I have written some codes to calculate the distance between the demons and the character and control the monsters' moving according to the space [12] (the professor said that the sqrt function would deal with much performance consumption). This idea was raised by Miles, who suggested we can make the monsters initially move randomly. If the distance between one of the monsters and the character is less than a specific range, the beast will begin to track the surface. The part of crude codes had a bug because Zun Yi did not code the direction precisely, which made the beast can only follow horizontally or vertically and recoding the law and the logic of the track function; this bug has been fixed successfully [13].

In addition, I learned how to make the character's movement more vivid and take a lot of time. Zun Yi did not know when he contacted this issue because he had never met this question. Finally, he created two lists to store two types of images, one of which was used to hold three different and continuous pictures which would look like moving when these pictures were shown one after another. Then, call these lists and travel them when the character moves to the right (R) or left (L), showing the character more dynamic.

In the end, the balance of in-game attributes is also a challenge. It's also essential to play a balanced game, ensuring it's not too easy but challenging. It's a straightforward game, but we wanted to get that balance right. Players and monsters have their initial attack power and health. After finishing each level, they can choose the attributes they want to enhance. The number of monsters in the next level will increase, and there will be a boss every five levels. That's pretty much what we started with.

DOI: 10.54254/2755-2721/30/20230071

Table 1. Comparing the algorithms used in the project.

Algorithm:	Collision algorithm	collision mask algorithm
	Less accuracy, and it also costs a lot of memory when the project is running.	More specific and accurate, as well as costs less memory.
Algorithm	track algorithm	baseline method.
	More efficient and advanced, consume less memory when the project run.	Consume a bunch of memory because of the sqrt method, and the uptime is increased.

3. Conclusion

With the gradual improvement of the project, our initial game design experience has come to a successful end. This is our first game design but not our last. It is a comprehensive application of the primary and professional knowledge we have learned and a test and enrichment of the knowledge we have learned. Though it has not done entirely because of the limited time, we decided to make our project looks like a game called soul knight; therefore, we did not add the end condition of the game and various weapon system, so the game like an unlimited circle without the highest scores and lack some important things. We use the algorithm we had learned in the lecture to make our program more sophisticated; we list and compare one from another in chart 1, such as the mask, to optimize the collection compared with the initial collection function, which can be more precise. It is a comprehensive process of re-learning and further improvement, cultivating our learning, independent thinking, and working ability. Through this game design, we also understand that learning is a long-term accumulation process, and we should continue to learn in the future work and life, strive to improve our knowledge and comprehensive quality, and solve theoretical or practical problems in the analysis and learning, to transform knowledge into practical training of ability.

References

[1] Guzsvinecz, T. The correlation between positive reviews, playtime, design, and game mechanics in souls-like role-playing video games. Multimed Tools Appl 82, 4641–4670 (2023). https://doi.org/10.1007/s11042-022-12308-1

[2] Yizhidagezi China. (2022) Pygame Introduction 2022 (4) Refactoring with Sprite classes. https://blog.csdn.net/qq_41068877/article/details/126495134?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216843263811680022741885 6%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168432638116800227418856&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-126495134-null-null.142^v87^koosearch_v1,239^v2^insert_chatgpt&utm_term=pygame%E7%B2%BE%E7%81%B5%E7%B1%BB&spm=1018.2226.3001.4187

[3] ZiYan Liu China.2022) Research on the construction of the worldview of role-playing games. https://kns.cnki.net/kcms2/article/abstract?v=3uoqIhG8C475KOm_zrgu4lQARvep2SAkueNJRSNVX-zc5TVHKmDNkpvkef-r_AczrjCLPnssc5F7MHQ2L7RFVXcjv4ASP-Tm&uniplatform=NZKPT

[4] Dong F.G. (2021) Introduction and practice to Python programming. Xidian University Press Xi'an

[5] Shang School China (2022) python+pygame teaches you how to implement tank battle in Python.https://www.bilibili.com/video/BV1wG411a74H/?spm_id_from=333.337.search-card.all.click&vd_source=b8c0d9577ecf4b206bc9b51cb4ca4ac5

[6] Xiaoyao Ma China (2022) python game library pygame classic tutorial.https://blog.csdn.net/weixin_63009369/article/details/127808805

[7] Wang, Z. (2021). Building Experiments with Pygame. Eye-Tracking with Python and Pylink (pp. 65-84). DOI: 10.1007/978-3-030-82635-2_3.

DOI: 10.54254/2755-2721/30/20230071

[8] Game_designer007 2022 A huge simple and easy-to-use pygame game pause and resume method https://blog.csdn.net/Game_designer007/article/details/105374871?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216841605531680018857211 5%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168416055316800188572115&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-105374871-null-null.142%5ev87%5econtrol_2,239%5ev2%5einsert_chatgpt&utm_term=pygame%E6%80%8E%E4%B9%88%E6%9A%82%E5%81%9C&spm=1018.2226.3001.4187

[9] Valente, A., Marchetti, E. Simplifying Programming for Non-technical Students: A Hermeneutic Approach. Künstl Intell 36, 17–33 (2022). https://doi.org/10.1007/s13218-021-00748-0

[10] Koupritzioti, D., Xinogalos, S. PyDiophantus maze game: Play it to learn mathematics or implement it to learn game programming in Python. Educ Inf Technol 25, 2747–2764 (2020). https://doi.org/10.1007/s10639-019-10087-1

[11] MISIR0927 2022 RogueGAME https://github.com/MISIR0927/RougeGAME

[12] dhjabc_1 2021 PyGame teaches you to implement a point-to-point intelligent tracking system step by step from 0 to 1 (one of them) https://blog.csdn.net/dhjabc_1/article/details/116589488?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216841601161680018852319 7%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168416011616800188523197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-116589488-null-null.142%5ev87%5econtrol_2,239%5ev2%5einsert_chatgpt&utm_term=pygame%E5%A6%82%E4%BD%95%E5%AE%9E%E7%8E%B0%E6%80%AA%E7%89%A9%E8%BF%BD%E8%B8%AA&spm=1018.2226.3001.4187

[13] dhjabc_1 2021 PyGame teaches you to implement a point-to-point intelligent tracking system step by step from 0 to 1 (two of them) https://blog.csdn.net/dhjabc_1/article/details/116650392?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216841601161680018852319 7%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168416011616800188523197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-4-116650392-null-null.142%5ev87%5econtrol_2,239%5ev2%5einsert_chatgpt&utm_term=pygame%E5%A6%82%E4%BD%95%E5%AE%9E%E7%8E%B0%E6%80%AA%E7%89%A9%E8%BF%BD%E8%B8%AA&spm=1018.2226.3001.4187