Testing Document
PacMan
version: 1

Group F8
1155127434 HO Chun Lung Terrance
Department of Philosophy, The Chinese University of Hong Kong
1155143519 WOO Pok
Department of Physics, The Chinese University of Hong Kong
1155157839 NG Yu Chun Thomas
Department of Computer Science and Engineering, The Chinese University of Hong Kong
1155157719 LEUNG Kit Lun Jay
Department of Computer Science and Engineering, The Chinese University of Hong Kong
1155143569 MOK Owen
Department of Mathematics, The Chinese University of Hong Kong

May 6, 2023

Contents

1 TEST PLAN

In order to test the software properly, we will apply a bigbang approach of software testing. We consider for each functionality of the software, there will be pure approaches (black- or white-box testing). To be specific, for complicated functionalities (mostly inter-module functionals and implementational functionals), we choose to apply black-box testing; for simple functionalities (mostly mono-module functionals and object reactions), we choose to apply white-box testing. Afterall, we will integrate the modules and test the whole system directly.

The testing will be under following environment:

- Software: Unity (version 2021.3.17f1), Windows7
- Hardware:
  - CPU: i5-8300
  - NVidia-GTX960M
  - 8GB-2400MHZ

Wewill use black box testing for: Menu UI, User Management, LeaderBoard, Shop system, Setting, Game Status, Player control, Ghost AI & Status

2 TEST CASES

2.1 Menu UI

For Menu UI, we will focus on the navigation between each UI panels under black box testing. There are totally 13 UI Panels in the application, at each instance only one panel will be displayed, and each panel takes up the entire window for the application.

In each panel there are buttons that allows us to either access another channel, go back to a previous channel, or quit the application. Here we will focus on testing the function of such navigation buttons.

Testing of buttons relating to user management (sign in, sign out, sign up, etc) will not be the focus here, but in the next section.

All 13 UI panels according to accessibility includes:

1. Freely accessable before starting a game: Login, Sign Up, Main,Level Selection, Settings, Leader Board, How To Play, Shop
2. Freely accessable during a game (without ending a game): Settings, In game and pause
3. Triggered only by gameplay status (death, victory): Game Over, Game Success and Respawn.

Inputs and expected outputs

1. Test Case 1.1: Login panel - Access.
   (a) Input: Opening the application.
   (b) Expected output: Login page displayed upon application start.
2. Test Case 1.2: Login panel - Quiting application.
   (a) Input: click the QUIT TO DESKTOP button in the Login panel.
   (b) Expected output: Application terminated.
3. Test Case 2.1: Sign up page - Access and qutting back to Login.
   (a) Input: click the SIGN UP button in Login, then click the GO BACK button in Sign Up.
   (b) Expected output: UI panel be switched from Login to Sign Up, then back to Login again.
4. Test Case 3.1: Main panel - Access by Login.
   (a) Input: click LOGIN button after filling with verfied email and password ("j5@j.com" and "55555555") in the Login panel.
   (b) Expected output: UI panel be switched from the Sign up panel to the Main panel.
5. Test Case 3.2: Main panel - Quit application.
   (a) Input: click the QUIT TO DESKTOP button in the Main panel.
   (b) Expected output: Application terminated.
6. Test Case 3.3: Main panel - Quit to Login by logging out.
   (a) Input: click the LOG OUT button in the Main.
   (b) Expected output: UI panel be switched from Main to Sign Up .
7. Test case 4.1: Level Selection - Access from Main and quiting back to main.
   (a) Input: click the START GAME button in Main, then click the BACK TO MENU button in Level Selection.
   (b) Expected output: UI panel be switched from Main to Level Selection, then back to Main again.
8. Test case 5.1: Settings - Access from Main and quiting back to main.
   (a) Input: click the SETTINGS button in Main; then clicking the DONE button in Settings.

   (b) Expected output: UI panel be switched from Main to Settings, then back to Main again.

9. Test case 6.1: How To Play - Access from Main and quiting back to main.
   (a) Input: click the HOW TO PLAY button in Main; then clicking the BACK TO MENU button in How To Play.
   (b) Expected output: UI panel be switched from Main to How To Play, then back to Main again.
10. Test case 7.1: Leader Board - Access from Main and quiting back to main.
    (a) Input: click the LEADERBOARD button in Main; then clicking the GO BACK button in Leader Board.
    (b) Expected output: UI panel be switched from Main to Leader Board, then back to Main again.
11. Test case 8.1: Shop - Access from Main and quiting back to main.
    (a) Input: click the SHOP button in Main; then clicking the GO BACK button in Shop.
    (b) Expected output: UI panel be switched from Main to Shop, then back to Main again.
12. Test case 9.1: In Game - Access from Level Selection - Scene 1 "OLD FACTORY", different difficulties.
    (a) Input: selecting 1 out of 3 difficulties but clicking EASY, NORMAL, or HARD, then click the OLD FACTORY button in Level Selection.
    (b) Expected output: UI panel be switched from Level Selection to In Game, starts the game with the game scene OLD FACTORY loaded, and cooresponding difficulty (number of ghosts and ghost AI intellegence).
13. Test case 9.2: In Game - Access from Level Selection - Scene 2 "ABANDONED MAZE", different difficulties.
    (a) Input: selecting 1 out of 3 difficulties but clicking EASY, NORMAL, or HARD, then click the ABANDONED MAZE button in Level Selection.
    (b) Expected output: UI panel be switched from Level Selection to In Game, starts the game with the game scene ABANDONED MAZE loaded, and cooresponding difficulty (number of ghosts and ghost AI intellegence).
14. Test case 10.1: Pause - Access from In Game and resuming back to In Game.
    (a) Input: pressing SPACE bar during a game, then click the RESUME button in Pause.

   (b) Expected output: UI panel be switched from In Game to Pause, game paused, then back to In Game again, with game resumed.

15. Test case 10.2: Pause - Jump to In Game by restarting.
    (a) Input: click the RESUME button in Pause.
    (b) Expected output: UI panel be switched from Pause to In Game, with the game restarted.
16. Test case 10.3: Pause - Jump to Settings and back to Pause.
    (a) Input: click the SETTINGS button in Pause, toggle different settings, then click DONE in Settings.
    (b) Expected output: UI panel be switched from Pause to Settings, with the game shown to be remaining paused in the background. Then UI switched from Settings back to Pause again.
17. Test case 10.4: Pause - Quit to Menu.
    (a) Input: click the QUIT TO MENU button in Pause.
    (b) Expected output: UI panel be switched from Pause to Menu, with the game ended.
18. Test case 11.1: Game Over - Access.
    (a) Input: Play a started game until losing (the character lost all health).
    (b) Expected output: UI panel be switched from In Game to Game Over.
19. Test case 11.2: Game Over - Back to Main.
    (a) Input: click BACK TO MENU button in Game Over.
    (b) Expected output: UI panel be switched from Game Over to Main, with the game ended.
20. Test case 11.3: Game Over - Restart.
    (a) Input: click RESTART button in Game Over.
    (b) Expected output: UI panel be switched from Game Over to In Game, with a new game started.
21. Test case 12.1: Game Success - Access.
    (a) Input: Play a started game until victory (all pallets are eaten)
    (b) Expected output: UI panel be switched from In Game to Game Success.
22. Test case 12.2: Game Success - Back to Main.

(a) Input: click BACK TO MENU button in Game Success.

(b) Expected output: UI panel be switched from Game Success to Main, with the game ended.

23. Test case 12.3: Game Success - Restart.

    (a) Input: click RESTART button in Game Success.

    (b) Expected output: UI panel be switched from Game Success to In Game, with a new game started.

24. Test case 13: Respawning - Access.

    (a) Input: Play a game so that Pacman dies (touches a ghost) without losing all health.

    (b) Expected output: UI panel be switched from In Game to Respawning briefly, then back to In Game again with Pacman respawned, and game continues from last death.

2.2 User Management

For UserManagement, we will focus on the black box testing on Login, Logout and SignUp. On the Login panel, there are 2 input fields: EMAIL and PASSWORDS for user input. With correct inputs, clicking The 'LOGIN' button should direct the user to the main menu. Else it should stay at the login panel.

On the SignUp panel, there are 4 input fields: USER NAME, EMAIL, PASSWORDS, CONFIRM PASSWORDS for user input. Where USER NAME should not be empty, EMAIL should with format of email, and PASSWORD should be the same as the CONFIRM PASSWORD. With valid inputs, clicking The 'SignUp' button should direct the user to the main menu. Else it should stay at the signUp panel.

On the main menu, the LOG OUT button should direct the user to the login panel and save the user data to the database. Assume the following accounts is stored in the database:

1. email: j5@j.com password: 55555555
2. email: test@gmail.com password: J1234567

Inputs and expected outputs

1. Test Case 1: Login with correct email and password.

    (a) Input: fill EMAIL with "j5@j.com" and PASSWORD with "55555555"

    (b) Expected output: Login Successfully with UserName Jest5 and switch to main menu.

2. Test Case 2: Login with correct email and incorrect password.

    (a) Input: fill EMAIL with "j5@j.com" and PASSWORD with "55555554"
    (b) Expected output: error occured with error message.

3. Test Case 3: Login with both incorrect email and password.
    (a) Input: fill EMAIL with "tes@gmail.com" and PASSWORD with "J1234567"
    (b) Expected output: error occured with error message.

1. Test Case 4: Login with correct email and other user's password.
    (a) Input: fill EMAIL with "test@gmail.com" and PASSWORD with "55555555"
    (b) Expected output: error occured with error message.

2. Test Case 5: Sign up with all valid input.
    (a) Input: fill USER NAME with "t1", EMAIL with "test5@gmail.com" and PASSWORD with "J5555555" and CONFIRM PASSWORD "J5555555"
    (b) Expected output: error occured with error message.

1. Test Case 6: Sign up with invalid username.
    (a) Input: fill USER NAME with "", EMAIL with "test5@gmail.com" and PASSWORD with "J5555555" and CONFIRM PASSWORD "J5555555"
    (b) Expected output: error occured with error message.

1. Test Case 7: Sign up with invalid password.
    (a) Input: fill USER NAME with "t1", EMAIL with "test5@gmail.com" and PASSWORD with "J555555" and CONFIRM PASSWORD "J5555555"
    (b) Expected output: error occured with error message.

1. Test Case 8: Sign up with empty confirm password.
    (a) Input: fill USER NAME with "t1", EMAIL with "test5@gmail.com" and PASSWORD with "J5555555" and CONFIRM PASSWORD ""
    (b) Expected output: error occured with error message.

1. Test Case 9: Logout.
    (a) Input: click LOG OUT button
    (b) Expected output: logged out and switch to login panel.

Results

The above results to be successful under the assumption of black box testing.

2.3 LeaderBoard

For LeaderBoard, we will focus on the black box testing on attempting adding the score to the leaderboard. If the score of the current player is higher than the minScore on the leaderboard, the leaderboard will update. Assume the current leaderboard is:

1. Cheater5 25380
2. Cheater6 25370
3. Cheater69 6969
4. Thomas D 6789
5. Min 1000

And the current username is "TEST".

Inputs and expected outputs

1. Test Case 1: Go to the leaderboard with score 1500
   (a) Input: click LEADERBOARD with score 1500
   (b) Expected output: replaces "5. Min 1000" with "5. TEST 1500"
2. Test Case 2: Go to the leaderboard with score 7000
   (a) Input: click LEADERBOARD with score 7000
   (b) Expected output: the last 3 places become:
       3. Min 7000
       4. Cheater69 6969
       5. Thomas D 6789
3. Test Case 3: Go to the leaderboard with score 99999
   (a) Input: click LEADERBOARD with score 99999
   (b) Expected output: the leaderboard becomes:
       1. TEST 99999
       2. Cheater5 25380

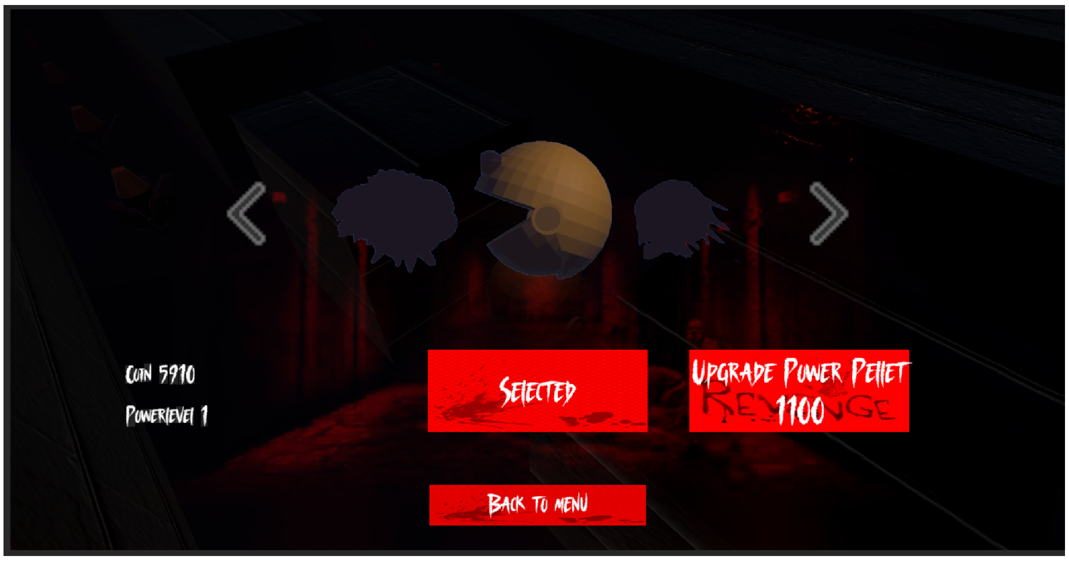
    3. Cheater6 25370
    4. Cheater69 6969
    5. Thomas D 6789

4. Test Case 4: Go to the leaderboard with score 87
    (a) Input: click LEADERBOARD with score 87
    (b) Expected output: No update.

Results

The above results to be successful under the assumption of black box testing.

2.4 Shop System



Under the assumption of black box testing, we focus on the functionalities of each button:

1. The ‘selected’ button should be changed to ‘buy’ or ‘select’ when the character focus is changed, and should be clickable to select the character.
2. The upgrade button should be changing the power level and coins and the value inside the button.
3. the ‘back to menu’ button should be able to direct player to the menu.
4. The left right button should beable to change the character shown.

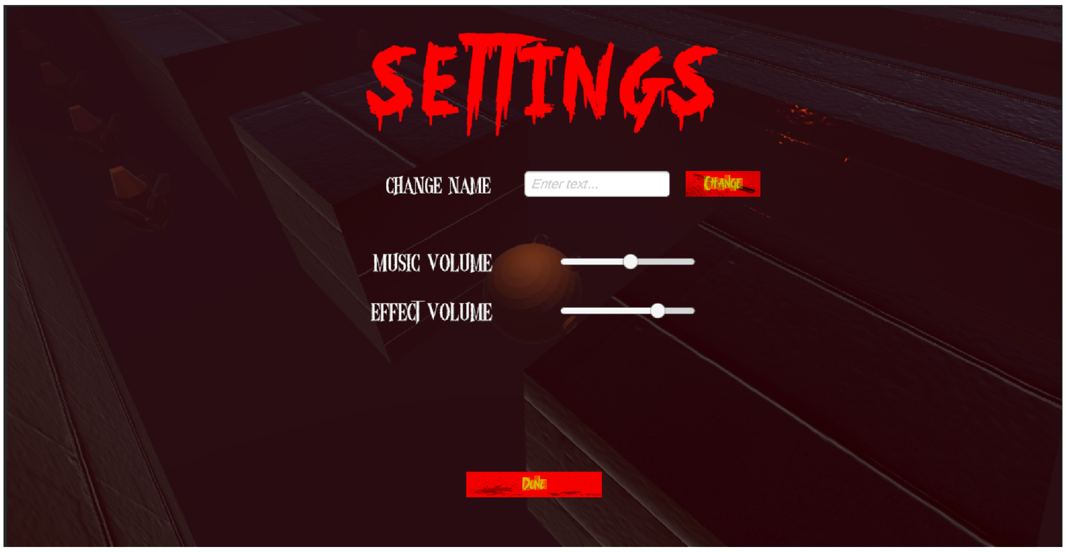
Inputs and expected outputs

1. Test Case 1: upgrade power pellet
   (a) Input: Clicking the upgrade button
   (b) Expected Output: coin decreased, powerlevel and value on the button increased.
2. Test Case 2: Clicking the back to menu button
   (a) Input: click the button
   (b) Expected Output: Back to menu
3. Test Case 3: Change to other character, and buy it.
   (a) Input: click left right button and buy.
   (b) Expected Output: character changed, and it will be successfully bought if coins enough, coins decresed, and button becomes selected; unsuccesful purchase will remain everything unchanged.
4. Test Case 4: Clicking blank space
   (a) Input: Clicking on blank area.
   (b) Expected Output: No output.

Results

The above results to be successful under the assumption of black box testing.

2.5 Setting



Under the assumption of black box testing, we focus on the functionalities of each button:

1. Name can be changed in the change name field, by clicking the change button.
2. Music and Effect Volume can be adjusted.
3. Done button will direct user to previous page.

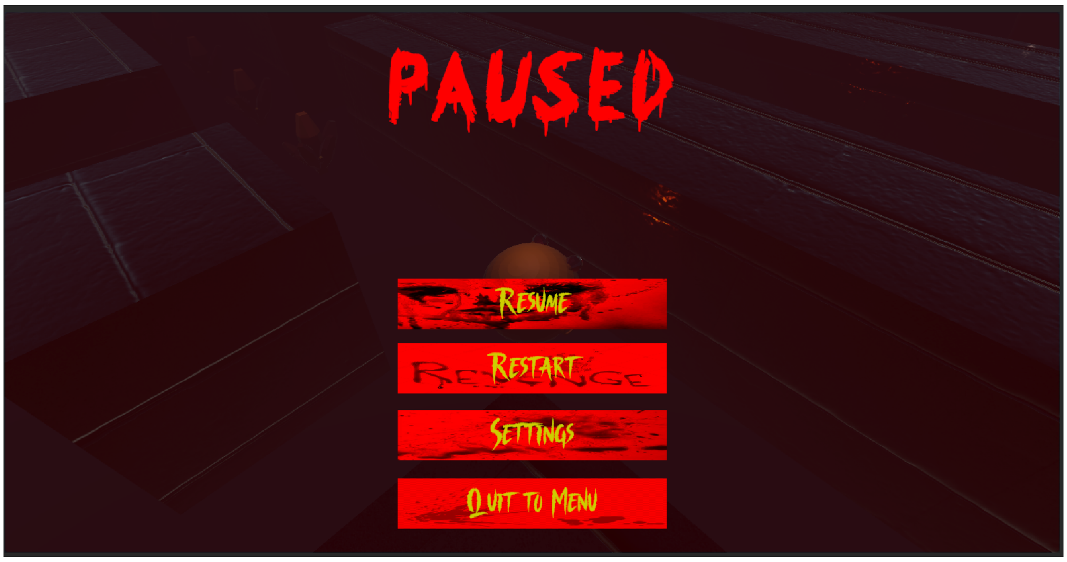
Inputs and expected outputs

1. Test Case 1: Directly clicking change button
   (a) Input: Clicking the change button
   (b) Expected Output: The name cannot be changed
2. Test Case 2: Input a new name and click the change button
   (a) Input: new name, change button
   (b) Expected Output: Name is changed.
3. Test Case 3: Adjust Music Volume to 0.
   (a) Input: adjust the slider of music volume
   (b) Expected Output: No music is played.
4. Test Case 4: Adjust Effect Volume to 0.
   (a) Input: adjust the slider of effect volume
   (b) Expected Output: No sound effect is played.
5. Test Case 5: Adjust Music Volume to nonzero.
   (a) Input: adjust the slider of music volume
   (b) Expected Output: music volume varies with slider.
6. Test Case 6: Adjust Effect Volume to nonzero.
   (a) Input: adjust the slider of effect volume
   (b) Expected Output: effect volume varies with slider.
7. Test Case 7: Clicking done button with different entrance
   (a) Input: Go to setting from different entrance, and click the done button
   (b) Expected Output: Returning the previous page.

Results

The above results to be successful under the assumption of black box testing.

2.6 Game Status



Under the assumption of black box testing, we focus on the functionalities of each button:

1. By clcking spacebar, we can enter the pause section
2. Resume button allows player resume the game
3. Restart button allows player restart the game
4. Settings button allows player enter setting page
5. Quit to menu allows player to halt the game

Inputs and expected outputs

1. Test Case 1: Never pressing spacebar during the game
   (a) Input: Usual game control
   (b) Expected Output: The pause section is never been called.
2. Test Case 2: Pressing spacebar anytime during the game
   (a) Input: press the spacebar
   (b) Expected Output: Pause section is called

3. Test Case 3: Clicking resume button
    (a) Input: resume button
    (b) Expected Output: Game continued.
4. Test Case 4: Clicking restart button
    (a) Input: restart button
    (b) Expected Output: Game restart
5. Test Case 5: Clicking settings button
    (a) Input: settings button
    (b) Expected Output: entering settings
6. Test Case 6: Clicking Quit to menu button
    (a) Input: quit to menu
    (b) Expected Output: diresting player to menu

Results

The above results to be successful under the assumption of black box testing.

2.7 Character's Movement and controls

For character movement control (both the player and the enemies), the main class of interest is the movement class controlling the player's action. We will be focusing on white box testing on the function SetDirection().

```
public void SetDirection(Vector3 targetDir, bool forced = false) {
    Debug.Log("SetDirection() called with target direction: " +
        targetDir.ToString() + " and forced: " + forced.ToString());

    if (isBlocked(targetDir) && !forced) {
        nextDirection = targetDir;
        Debug.Log("Path is blocked. Setting nextDirection to: " +
            nextDirection.ToString());
    }
    else {
        nextDirection = targetDir;
        direction = targetDir;
        nextDirection = targetDir;
        Debug.Log("Path is not blocked. Setting direction and nextDirection
            to: " + direction.ToString());

        rotation = new Vector3(0, Mathf.Atan2(direction.z, -1 * direction.x) *
            Mathf.Rad2Deg - 90, 0);
        if (rotation.y < 0) rotation += new Vector3(0, 360, 0);
        if (rotation.y >= 360) rotation -= new Vector3(0, 360, 0);
        // Force rotation.y to become 0 or 90 or 180 or 270
        rotation = new Vector3(0, Mathf.Round(rotation.y / 90) * 90, 0);
        Debug.Log("Calculated rotation: " + rotation.ToString());
    }

    // Add this debug line to log the current direction, nextDirection, and
        rotation
    Debug.Log("Current direction: " + direction.ToString() + ", nextDirection:
        " + nextDirection.ToString() + ", rotation: " + rotation.ToString());
}
```

And here is the function update() in movement.cs who take care of the variable NextDirection.

```
private void Update() {
    if (nextDirection != Vector3.zero) {
        SetDirection(nextDirection);
    }

    // Add this debug line to log the current nextDirection value
    Debug.Log("Current nextDirection: " + nextDirection.ToString());
}
```

Here we added debug message for checking the validity of the output direction sequence.

Inputs and expected outputs

1. Test Case 1: Pacman moves in an open path
    (a) Input: targetDir = Vector3.forward (0, 0, 1)
    (b) Expected Output: direction and nextDirection should be set to Vector3.forward, and the appropriate rotation should be calculated.
2. Test Case 2: Pacman moves in a blocked path without forced movement
    (a) Input: targetDir = Vector3.left (1, 0, 0) and a wall in the left direction

   (b) Expected Output: nextDirection should be set to Vector3.left, but direction should not change.

3. Test Case 3: Pacman moves in a blocked path with forced movement

   (a) Input: targetDir = Vector3.left (1, 0, 0), a wall in the left direction, and forced = true

   (b) Expected Output: direction and nextDirection should be set to Vector3.left, and the appropriate rotation should be calculated.

4. Test Case 4: Pacman changes direction when a previous blocked path becomes unblocked

   (a) Input: Pacman is moving forward (0, 0, 1) and attempted to turn left (1, 0, 0) but it was blocked. The wall blocking the path is now removed.

   (b) Expected Output: direction and nextDirection should be set to Vector3.left, and the appropriate rotation should be calculated.

Outputs and analysis

Here are the output log messages:

1. Test Case 1: Pacman moves in an open path

```
SetDirection() called with target direction: (0, 0, 1) and forced:
    False
Path is not blocked. Setting direction and nextDirection to: (0, 0, 1)
Calculated rotation: (0, 0, 0)
Current direction: (0, 0, 1), nextDirection: (0, 0, 1), rotation: (0,
    0, 0)
```

2. Test Case 2: Pacman moves in a blocked path without forced movement

```
SetDirection() called with target direction: (1, 0, 0) and forced:
    False
Path is blocked. Setting nextDirection to: (1, 0, 0)
Current direction: (previous direction), nextDirection: (1, 0, 0),
    rotation: (previous rotation)
```

3. Test Case 3: Pacman moves in a blocked path with forced movement

```
SetDirection() called with target direction: (1, 0, 0) and forced:
    True
Path is not blocked. Setting direction and nextDirection to: (1, 0, 0)
```

```
Calculated rotation: (0, 270, 0)
Current direction: (1, 0, 0), nextDirection: (1, 0, 0), rotation: (0,
    270, 0)
```

4. Test Case 4: Pacman changes direction when a previous blocked path becomes unblocked

```
Current nextDirection: (1, 0, 0)
SetDirection() called with target direction: (1, 0, 0) and forced:
    False
Path is not blocked. Setting direction and nextDirection to: (1, 0, 0)
Calculated rotation: (0, 270, 0)
Current direction: (1, 0, 0), nextDirection: (1, 0, 0), rotation: (0,
    270, 0)
```

The test cases are successful. Furthermore, simple black box testing (on modules other than this one, or say the overall movement ability) was also done when testing this current module. We conclude that we did not discover significant faults on the character movement feature.

2.8 Big-bang

We integrate the modules and test the whole system once. The procedure will be as follows:

1. Start the program. Sign up a new account and Logout once, and login and Go to settings and adjust volumes.
2. Return main menu and enter shop, checking the initial status of equipment.
3. Return main menu and start a new game, choose random level and venue.
4. During the gameplay, press spacebar to call the pause menu at least once to Check all buttons in the pause menu.
5. When the current game is finished, go to scoreboard and shop to check the change of status.
6. Go to Setting and chenge a new name.
7. Repeat 3-6 for several times.
8. Logout the game and then Login the game after some time, and check all game status and settings.
9. Go to scoreboard to clear all records and Quit the game directly from the main menu.

Results: No bugs have been detected. More functionalities will be added in the future.