# M3 – Unit test write up

## Player Collision Test

Description for PlayerCollisionTest.java:

This class contains two unit tests:

My unit tests verified the collision physics with solid objects containing convex polygon “wireframes”. I tested head-on collisions as well as “indirect” collisions where some of the player’s velocity is preserved in the direction parallel with the side of the object the player collided with. My code verifies that basic collisions work, as the player is stopped in only the direction that would cause them to collide with the object.

1. testZeroVelocityAfterDirectCollision()

In testZeroVelocityAfterDirectCollision(), the program applies a constant force to the player object after placing a solid wall immediately left of the player. The force matches that which would be applied when the player holds down A during a game. The game’s physics is then updated 10,000 times, and the player’s velocity is measured and verified to be 0. So, the player collided with the object and could not pass it.

2. testSlidesAfterPartialCollision()

In testSlidesAfterPartialCollision(), the program creates a vertical line of walls, and places the player to the right of it. A large but temporary leftwards and downwards force is applied to the player object. The game’s physics is then updated 500 times, and the player’s x and y position are measured to be underneath where it started yet immediately to the right of the wall. So, the player slid downwards, parallel in the direction of the wall and in the direction of the applied force.

## Next Room Test

Description for NextRoomTest.java

This class contains 3 unit tests:

1. fourExitInStartTest

checking if the initial room has exactly 4 exits.

2. nextRoomTest

By setting the player’s position to one of the exits in the initial room. Then, wait for one second for the system to more the player to the next room. After that, if the id of the room in which the player locates currently is different from the starting room id, then player is in a different room.

3. finishGameTest

By setting the room to 999 ( special room id for the final room), it put the player in the final room. when it is in the final room, there will be a button for it to exit the game and go to the finish page.

# Room Layout Test

Description for RoomLayoutTest.java.

This class has 3 unit tests:

1. testGameStageRoom()

This test checks that the initial room the player enters has the appropriate design layout by checking that the design corresponds to expectation based on the room ID.

The test then verifies that the room layout is within the boundaries of the room, so no tiles are outside the walls. The test does so by getting all the physical objects. It then checks that the “wall tiles” are within the bounds.

2. testDesignID()

The test checks that the layout design will be appropriate for any room.

It does so by checking that the room ID from one to thirty all create the appropriate layout.

This ensures that the layout is specific to the room, and will not change when the player leaves and returns to a particular room.

3. testDimensionWall()

This test checks that if the room’s width and height changes, the tiles will still behave as expected.

This does so by checking when the width is fixed, and the height increments from 20 – 100. After each increment, the “wall tiles” are checked to be within the boundary.