The player simulation type thar I have chosen is a platformer. The simulation ends when the player has reached the platform on the right side of the screen. The player moves by pressing the A and D keys, it jumps by pressing the W key. The player shoots bullets but clicking.

I used my own Vec2 class in which I implemented all the necessary methods. Some of the methods in the Vec2 class reuse code by calling another method (the methods for rotation, for example run the code only once for radians, the other method coverts the degrees to radians and calls the other method). The methods in the class are tested in the TestVectorFunction class.

There are two types of aiming. Aiming in the current direction and aiming to a target are both implemented. On the platforms there are turrets that shoot in the current direction, while the player shoots to a target. There are also two types of bullets: both the player and the turrets shoot small bullets which are not influenced by gravity and big bullets (shot only by the player when the left shift key is pressed). The small bullets are destroyed when colliding with something, while the big bullets bonce a couple of times before getting destroyed. The turrets can be destroyed only by the big bullets.

The simulation uses ball-ball collision and ball-segment collision. The player movement is influenced by the platform's bounciness (coefficient of friction). The platforms simulate ice, sand and a normal ground. This makes the player either move faster or slower on the platforms and makes stopping either longer or shorter.

When the player collides with a bullet, the player will be pushed back by an impact force. The impact force takes the direction of the normalized velocity and multiples it by a certain amount. When the player reaches the big gap, the gravity will be inverted, simulating some sort of wind force that pushes the player up. The player has to lower the stairs in order to move up. The player needs to shoot a small bullet at the stairs ball to lower them. When the ball is hit, it disappears and one of the ends of the segment is rotated until it gets into place.

The simulation creates the platforms by using the class Platform. In it the platforms are added to the list of lines and are also removed if the line is destroyed. Because the player and bullets work in different ways, I included the code for collision detection and resolve in both classes. This way there can extra functionality added to either the player or the bullet without encountering any bugs.

Challenges encountered:

- Code structure: the code structure was a little challenging to do. At first both the player and the bullets extended the same class. Because the player and the bullets behave differently, I ran into some bugs that were fixed by separating the two classes and writing separate code for each of them.
- Collision with the ball placed at the end of the segment: the balls at the end of the segments are in fact bullets that are not moving. When the player collided with these balls it applied the impact force to the player. To fix this I created another variable that stated what the ball is - a bullet or a segment ball.
- Movement on the platforms: this is the first thing I started to work on. In order to make sure that the player cam move properly on every platform I created a variable that stores the current platform that the player is colliding with. The velocity takes the current platform direction and multiplies it the speed variable which is increased or decreased when the player presses A or D. Another issue was related to the orientation of the platforms. If the normal was pointing down the movement needed to be inverted.