1 Overall Design

Quality: Excellent

There are three packages in the source code.
Core contains the core models in the game, *Display* contains all the classes relating to user interface, including the main class of the program. *ScoreData* contains a class that can read a file and save a new high score to the txt file that contains data of scores. The images and sound effects are stored in separate files. Each class is used only for reasonable use and has good cohesion. There remain sensible consistency between each classes and methods. The methods are placed in the order regarding different functionalities and different models. Names of variables and methods followed the naming conventions of java.

2 Input

Quality: Excellent

User input is completed and all actions are implemented.

1. In the game screen:
   The player can press the *left/right arrow* to rotate the ship.
   The player can press the *space bar* to fire bullets.
   The player can press the *upper arrow* to apply thrust.
   The player can press *"h"* key to apply hyper space jump.
   The player can press *"x"* key to exit during game play.
   The player can press *"b"* key to apply brake when lost control of the speed.
   The player can press *"p"* key to pause and continue the game.

2. In the menu screen:
   The player can press *"n"* key to start new game.
   The player can press *"a"* key to see the info (controls) of the game in the about screen.
   The player can press *"h"* key to see the high scores in the score screen.
   The player can press *"x"* key to exit the program.

3. In about screen and score screen:
   The player can press the *"m"* key to get back to the menu screen.

3 Display

Quality: Excellent

All required objects are drawn on the screen, which includes the player ship, player ship thrust, player shield, three sized asteroids, player bullets, enemy ship, enemy shots, enemy shield, enemy reset times, ship explosions, smallest asteroid explosion, life bonus, shield bonus and score bonus. In the game screen displayed in Figure 1, Asteroids, player ship, player ship thrust, enemy ship and a shield bonus is shown on the screen.

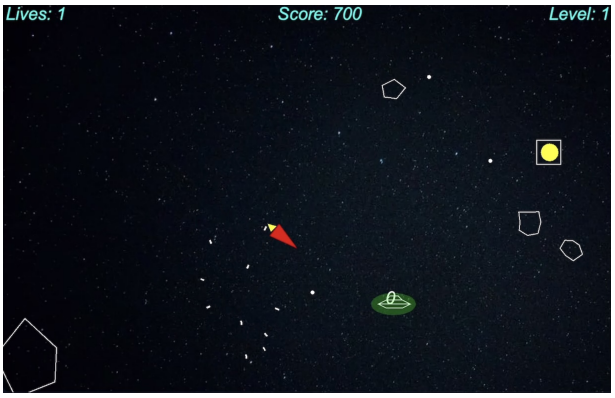

Figure 1: Game Screen

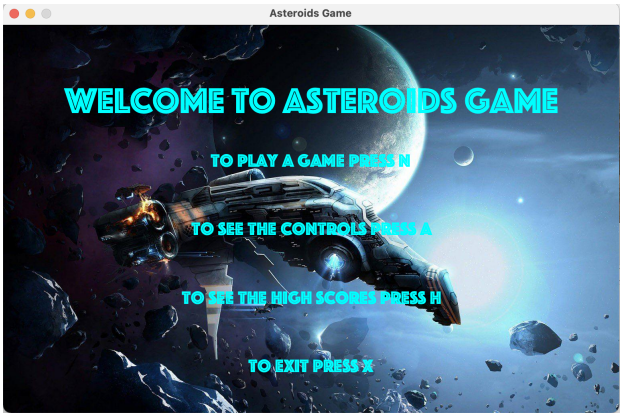

Figure 2: Main Menu Screen

4 Menu

Quality: Excellent

The game contains a *menu screen* (Figure 2), a *hall of fame* (Figure 3) display and an *about screen* (Figure 4) with information of game controls.

Figure 3: Score Screen

Figure 4: About Screen

5 High Scores

Quality: Excellent

There is a class called *ConsistentScoreKeeper* in the code that implements the *ScoreKeeper* from the provided engine. There is a txt file in the same file as the program that preserves the top ten scores in the rank of the game. The game would only ask for the player's name when his/her score is greater than the lowest score in the rank. And the score will be stored in the txt file. It should be noted that multiple high scores of a same player (same name) can be stored in the rank. Different players with same scores can also be in the rank.

Moreover, when the player's score reaches a multiple of 10000, a free life is given.

```
private void generateRandomShape() {
    Polygon randomAsteroidShape = new Polygon();
    int f = 5 + (int) (5 * Math.random());
    double da = (2 * Math.PI) / f;
    double a = (2 * Math.PI) * Math.random();
    for (int i = 0; i < f; i++) {
        double ad = halfSize + halfSize * Math.random();
        double ax = ad * Math.cos(a);
        double ay = ad * Math.sin(a);
        randomAsteroidShape.addPoint((int) ax, (int) ay);
        a += da;
    }
    shape = randomAsteroidShape;
}
```

Figure 5: Generate random shape for asteroids

```
private void createBonus() {
    if (playerScore >= (c + cc) * bonusScore) {
        cc++;
        if ((int) (6* Math.random()) == 0) {
            int type = (int) (3 * Math.random());
            bonuses.add(new Bonus(type));
            c++;
        }
    }
}
```

Figure 6: Create bonus randomly

6 Randomness

Quality: Excellent

1. Asteroids: The asteroids are originally created in the edges of the screen, but in random *position*. Their *velocities* for x,y coordinate and rotation and the movement *angle* are also randomly created. After the biggest and medium sized asteroids are hit, the directions of the newly created smaller asteroids are randomly selected in the required range. As displayed in Figure 5, the shape of every asteroid is randomly generated.
   The sparks for the explosion of the smallest asteroids last for random amount of time.
2. Enemy Ships: The size, original position and the position to move toward (within a certain range) of the enemy ship are randomly generated and its next shot time and next show time are also chosen randomly.

3. Bonus Boxes: For Bonus boxes, the position and the next position to move to are randomly generated. As shown in Figure 6, there is one-sixth possibility for the bonus to be created in 100 scores, and the type of the bonus is also randomly chosen.

7 Alien Ship

Quality: Excellent

1. As described in the previous section, enemy ships appears, moves and shoot bullets randomly. It can also avoid collision with asteroids while moving across the screen. The code is displayed in Figure 8.

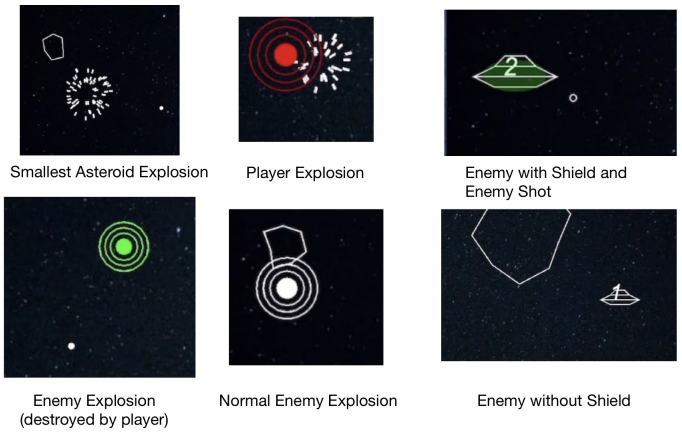

Figure 7: Various displays in the game

```
public void move(){
    double dx = targetX - x;
    double dy = targetY - y;
    double dist = Math.sqrt(dx * dx + dy * dy);
    if (game.checkEnemy(targetX,targetY,level,size)) {
        vx += (dx / dist) * 0.1;
        vy += (dy / dist) * 0.1;
        vx = vx > 2 ? 2 : vx < -2 ? -2 : vx;
        vy = vy > 2 ? 2 : vy < -2 ? -2 : vy;
        x += vx;
        y += vy;
        screenWrap();
    } else{
        resetTarget();
        move();
    }
}
```

Figure 8: Enemy ship avoid asteroids code

8 Bonuses

Quality: Excellent

1. Explosions:
   As displayed in Figure 7, when the smallest asteroids are destroyed, there will be sparks on the screen in the position of the asteroid and last for random amount of time.
   When ships are destroyed, there will be ships explosions on the screen in the same position. For enemy ships it could be green (hit by player bullet) or white and for player ship it is red. The explosions scale of enemy ships relates to the size of the ships.

2. Bonus Boxes:
   There are three kinds of bonus boxes in the game, as displayed in figure 9. The figure 6 displays the code of random creation of boxes and the details are described in the *Randomness* section. It's written in the assignment document that the bonuses should be randomly dropped when an asteroid is destroyed. However, in my opinion one-sixth percentage in every 100 scores can also meet this need.
   The *life bonus* gives an extra life to the player; the *shield bonus* gives a shield to the player ship that can defend any form of attacks; the *score bonus* gives 200 points to the player. The bonus boxes won't follow the player to the next level.

3. Player Ship Thrust: As displayed in Figure 1, when thrust is applied, the thrust is also drawn on the screen in yellow triangle. When the player keep pressing it (thrust is continuously applied) the thrust can blink in every three counts of thrust.This is achieved through getting the modulus of the count divided by 3.

4. Background Images:
   The back ground images for menu, about, score and game screens are added through
   "BufferedImage myPicture = ImageIO.read(new File("Images/xx.jpg"));".

5. Sound Effects:
   Sound effects are added when player fires an bullet, player apply thrust, player apply hyper space jump, player explosion, enemy appears, enemy shoots a bullet, player pick up bonus boxes, player picked up different kinds of bonuses, a level is completed and a new game is started.

Even some methods throw exception, but they are all caught in the methods that use them with try-catch statement.

6. Player Shield:
   When a new game starts, or the player is reseted after being destroyed or the player is moved to the next level, a shield is created for a short amount of time. The time is ten times longer when it's a bonus.

7. Enemy Shots:
   The bullet of enemy is created by different class called "EnemyShots", the shots works like guided missile, they can follow the player ship and can be destroied by player bullets.

8. Enemy Ships Rules:

   - One enemy is created in each level, its "set up times" (how many times they'll be coming back with the same shape) equals to the level number, the number is drawn on the center of the ship.
   - If it has not been destroyed completely in the previous levels, it will follow the player to the next level. Player only has to destroy all the asteroids to get to the next level.
   - As displayed in Figure 7, if it's destroyed by the players bullet, the explosion will be green, otherwise it's white.
   - As displayed in Figure 7, the enemy ship has a shield that can protect them only from the asteroids when they first appear on the screen, the time is "150 + level*100", which means the shield will last longer as the level increases.

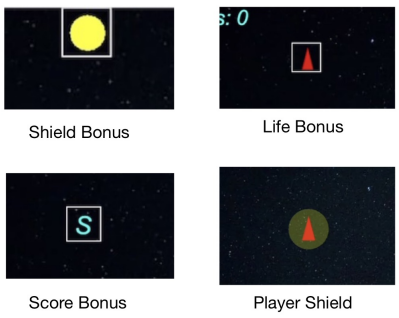

Figure 9: Bonus boxes and Player shield

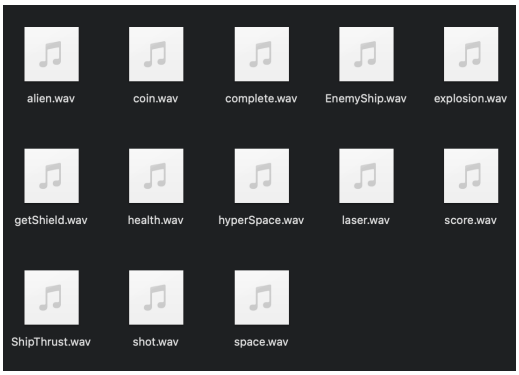

Figure 10: Sound effects

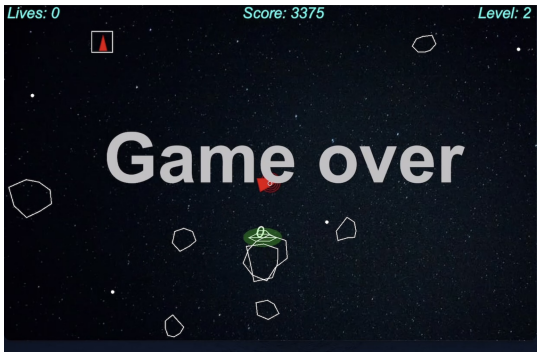

Figure 11: Game Over

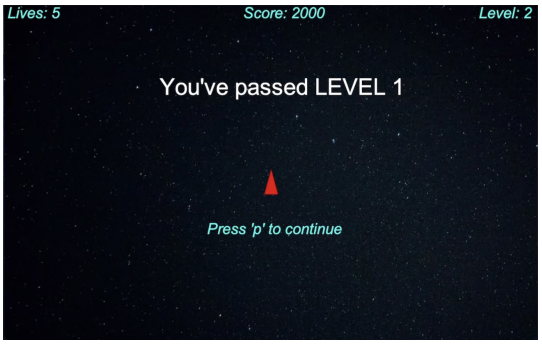

Figure 12: Passed a Level

9 Levels

Quality: Excellent

Levels can increase infinitely until the player looses. The player is told that he/she has passed a certain level after completing a level. (Figure 12)