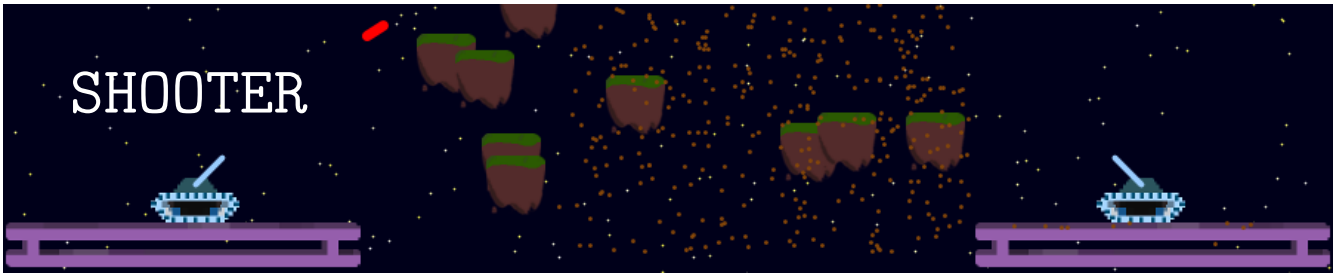
Report — CS4303 Practical 1

140013112

October 3, 2018



Instructions to run game:

```
$ java -jar Shooter.jar
```

Introduction Screen:

Use point and click to select `1vs1` (Human vs Human) or `Story` (Human vs AI).

In Game:

5 games will be played for either mode. The dots in the top-left corner indicate the progress through the 5 games. Player 1 is the left player, player 2 is the right player. The active player is indicated by a red triangle above the tank. The active player must destroy the opponent with a shot.

Controls:

The tank may be moved with `a` – left, `d` – right. The gun angle is adjusted with `w` – up and `s` – down. The force is increased with `r` and decreased with `f`. `space` fires the gun and passes on the turn. Limits have been set for force, angles and movement.

Code:

The submission includes a compiled jar, the report pdf, and the source code in the `src` folder. In the `src` folder, the game is started off with the `Game/MainGame.java` file.

1 Overview

For this practical we were asked to implement a variant of the classical Artillery Game in Processing. The main game elements identified were – 2D space, 2 tanks, arced shots and terrain features.

I chose to name my game "Shooter" and create a science-fiction artillery variant set on a foreign planet with floating rocks. The background simulates deep space. Tanks drive on metal platforms. Blocks of rock may be shot and explode into a cloud of sand. A short background story is included when playing the AI. All graphics were hand-drawn.

2 Design

The main challenges faced during the implementation of this game was making the game immersive and (relating to that) increasing the physics realism.

2.1 Game Structure

The game was divided into multiple stages which are represented by the enum `State` in `GameState.java` with their order shown in figure 1. The `Story` states are only used in the Story Mode.

The game starts and ends with an introductory screen, explaining the controls and allowing the user to select the game mode. Each duel (set of 5 confrontation) starts with player 1 (left player), then each player gets a turn. If player 1 hits player 2, player 2 will have the first shot in the next confrontation. This increases fairness.

2.2 Graphics and Controls

Tank movement was restricted to given platforms to increase the duel-like feel. It also limits the available actions to the user, making users focus on using wind and gravity to shoot shells at opponents. Gun angles and forces are also restricted as to reduce the accidental risk of shooting oneself, however that is still a risk.

Blocks move randomly up and down to simulate floating. As blocks are not influenced by gravity (because they float), I chose not to take wind into account for the blocks either and regard them as "fixed" terrain. Wind is visualized by blocks randomly shedding sand particles. Sand particles may land on platforms, however ignore blocks and tanks.

Block placement and number is random for each confrontation, however it is made sure that at least 5 blocks are placed near the bottom of the terrain area. This stops players from shooting straight at each other with high power.

Wind is randomly decided (within limits) for each confrontation, however stays constant for the duration of the confrontation. This allows players to adjust to the wind for the duration of one confrontation. Gravity is fixed.

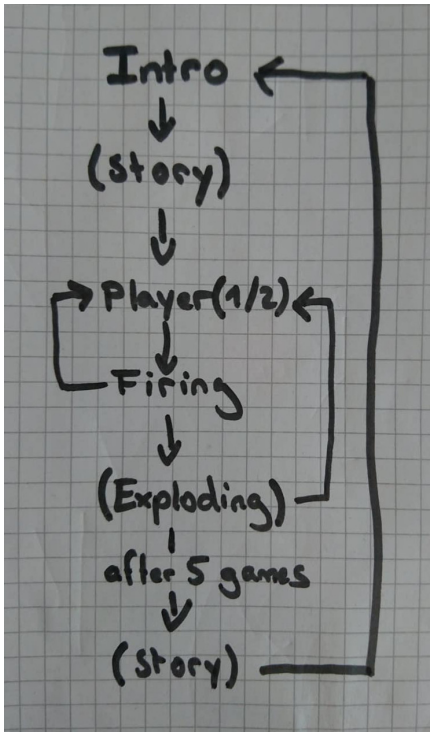


Figure 1: Game States

All movable game elements (shells, sand) are removed when they leave the right, left or lower bounds of the game screen. Objects leaving the upper bound are not affected, allowing for high shots.

2.3 Physics

Only sand particles and shells are influenced by wind and gravity. They make use of the force accumulators and dampening discussed in the lectures. For each frame, all forces acting on the particle are taken into account and integrated to find the resulting position and velocity.

All particles have a starting velocity. For the shell this is defined by the user-chosen force of the shot. Sand particles either have the wind force as the starting velocity or the the colliding shell's velocity if a block was hit by a shell. Sand particles on platforms only retain their x velocity and have a higher dampening factor, to simulate being pushed by the wind along metal with higher friction.

2.4 AI

An AI player was implemented as part of the story mode of the game. The AI is able to move and adjust gun values to hit the human player. Currently the AI player is always player 2 (on the right).

The AI has 5 states as follows: `deciding move` → `moving` → `finding the best force and angle` → `adjusting gun values to the desired values` → `firing`.

All states are executed in different frames and states may span over multiple frames. This was done as to not hang the game while the AI decides on its actions. It also allows the AI to move in a more human-like manner and allows the user to track the AI's changes to the gun.

2.4.1 Decision Making

Tank movement is decided at random and will at most be 30 pixels in any direction.

To choose gun values, the AI will loop and produce random angles and forces (within a sensible range). For each angle and force combination, a shell is simulated without regarding blocks. The shell's minimum distance to the opponent's center is measured for each simulation. If the shell lands within the explosion range of the opponent plus a random tolerance determined by the progress through the game, the force-angle combination is chosen as the desired configuration. The random tolerance is initially 15, and decreased by 3 after each confrontation. After each firing, a new configuration is determined, as the opponent may have moved or the AI might have missed or hit a block.

This random approach was taken, as shell arcs are very hard to calculate given wind, drag and gravity forces acting upon it. Furthermore, there are often many configurations which may hit the opponent, and a "fire - observe - adjust" approach would always aim for a predefined "best" shell arc, even though the developer may not know about trick shots. This random AI, for example, taught me how to shoot straight up with a high force, allowing wind and gravity carry the shell over the blocks onto the opponent.

AI difficulty is balanced, as the AI does not take blocks into account, will not necessarily make the same shot twice (even though a block might be destroyed) and adds a random tolerance to the shell distance to the opponent. The AI will also adjust the gun values in steps of the same resolution as human players would. This means the AI will never have exactly the gun values it desires. An experienced player can easily beat the AI while still taking 1 or 2 hits.

When testing my game, fellow students generally lost to the AI in their first duel and won in their second duel.

2.5 Story and Setting

The setting is that of a dystopian future on an alien planet in the deepest part of space. The duel is happening at a high altitude on that planet with night sky in the background. The floating rocks are not explained in the game due to time constraints when writing the story for the story mode.

The story was designed to adapt to the events in the game. Again, due to time constraints, the story was limited to 2 screens, with the latter screen being influenced by the player's win or loss.

3 Implementation

This game was implemented in Java using the Processing library. The IntelliJ IDE was used rather than the Processing IDE. This is the reason for `.java` file-endings. Also an instance of the `PApplet` must be passed to each object which wants to make use of Processing functions like drawing. And the game must be started through a `public static void main(){}` method.

3.1 Code Structure

The code files are divided into the packages `Game`, `Game Objects` and `Screens`.

`Game` includes files related to the structuring of the turns, loading of game elements and screens and keeping track of the current game state. The `GameState` class is a singleton class as this is a global state which can only exist once. It stores the history of the states in an ArrayList which keeps track of the player's turn, if the player has already fired, etc. It also keeps track of player scores, the position in the story and difficulty of the AI.

`GameObjects` includes code files relating to individual objects in the game. The physics code was also included in here.

The `Screens` package has the background, introduction and story screens.

3.2 Game Elements

The `ForceRegistry` was altered from the lecture slides to allow for multiple `ForceGenerator`s to be registered to a single particle. All the registrations were then stored in a `HashMap` with the particle as the key. This allows fast access to registrations and fast deletion of sand particles from the registry.

Buttons are implemented by drawing rectangles and text, checking if mouse pointer position was over the rectangle and accepting click events.

User input is handled using the `keyPressed()` methods. Processing will only fire the user input methods in the `MainGame` file. The code there will then forward the events on to the correct screen/duel which handle the events separately.

PNGs were used for all graphics as they may include transparency. However, all tanks and blocks are still regarded as rectangles when it comes to collision detection.

Figure 2: Introduction Screen

4 Conclusion

In conclusion, the presented game implementation satisfies all of the basic requirements and shows some extensions. It provides a modern science-fiction take on the classic artillery game. The game aims to immerse the player into a story and setting, makes use of some physics and allows the user to play a fair and fun game against other users or an AI.

4.1 Possible Improvements

If time allowed, the game could have been extended to include different seasons (changing appearance of graphical game elements). The story could also have been extended to interleave with the gameplay and be influenced by the performance of the player. The use of realistic physics could also have been extended to allow explosive forces to act on surrounding blocks or platforms. Also some difficulty adjustment for the AI would have been nice.

5 Appendix

Screenshots are provided to illustrate gameplay.

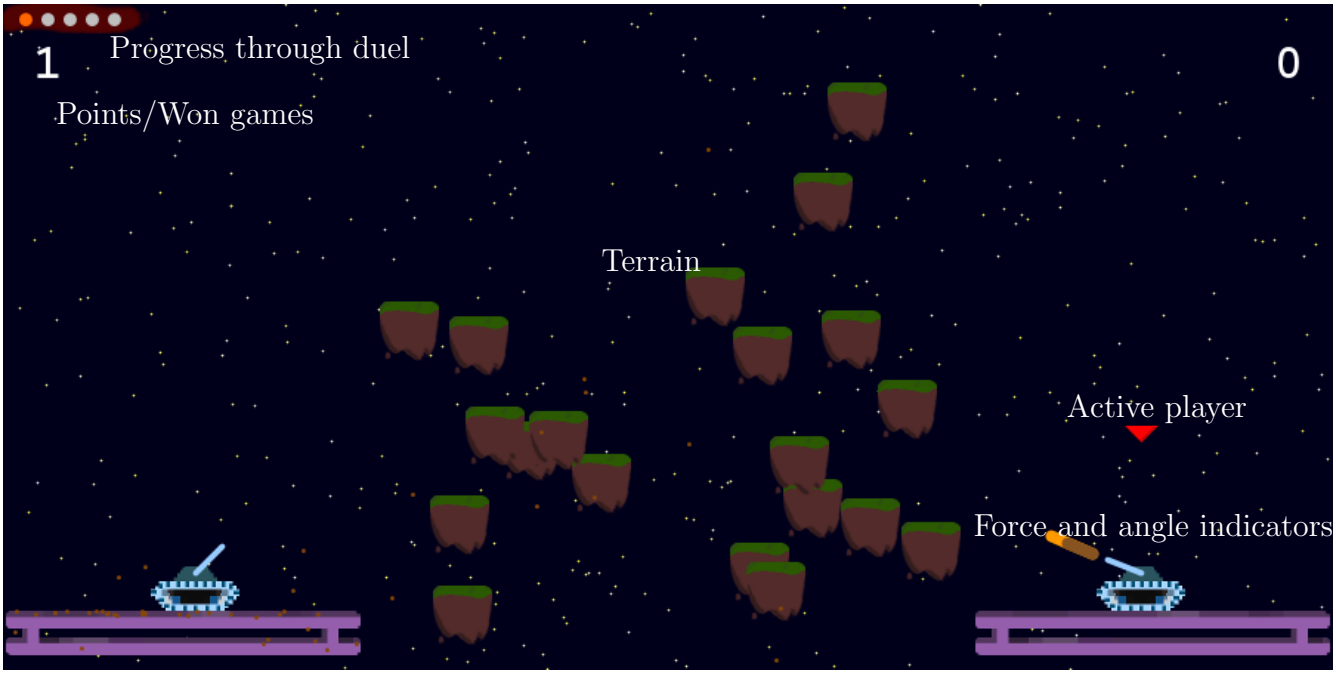


Figure 3: Confrontation 2/5

Figure 4: Story Screen for Story Mode