# CS4303 P1 - Artillery

220026989

## 1. Overview

The aim of this practical was to create a game that utilizes a basic physics engine
to model projectile motion. The game has 2 controllable tanks and both of the tanks have the ability to move along the terrain. Each tank has a cannon with which it can aim and fire the cannonball (bullet) that can potentially destroy landscape or deal damage to the enemy tank.
The game further contains the notion of gravity, airdrag (friction) and wind which alters the trajectory of the cannonball. Furthermore both of the tanks are considered as real world objects to a certain extend, and has the ability to bounce when falling. Additionally both the tanks get two abilities :

a) Gravity gun
b) Landslider
c) FlashBomb

The gravity gun changes the gravity of the terrain, while the landSlider clears a whole column of land in front of the player whilst sacrificing a bullet shot. FlashBomb as the name suggest flashes the other player for his next move. The opposing player will be completely blind for his next move. The flashBomb player keeps the players in their toes as, brilliant use of the same could change the tide of the battle.

## 2. Playing techniques

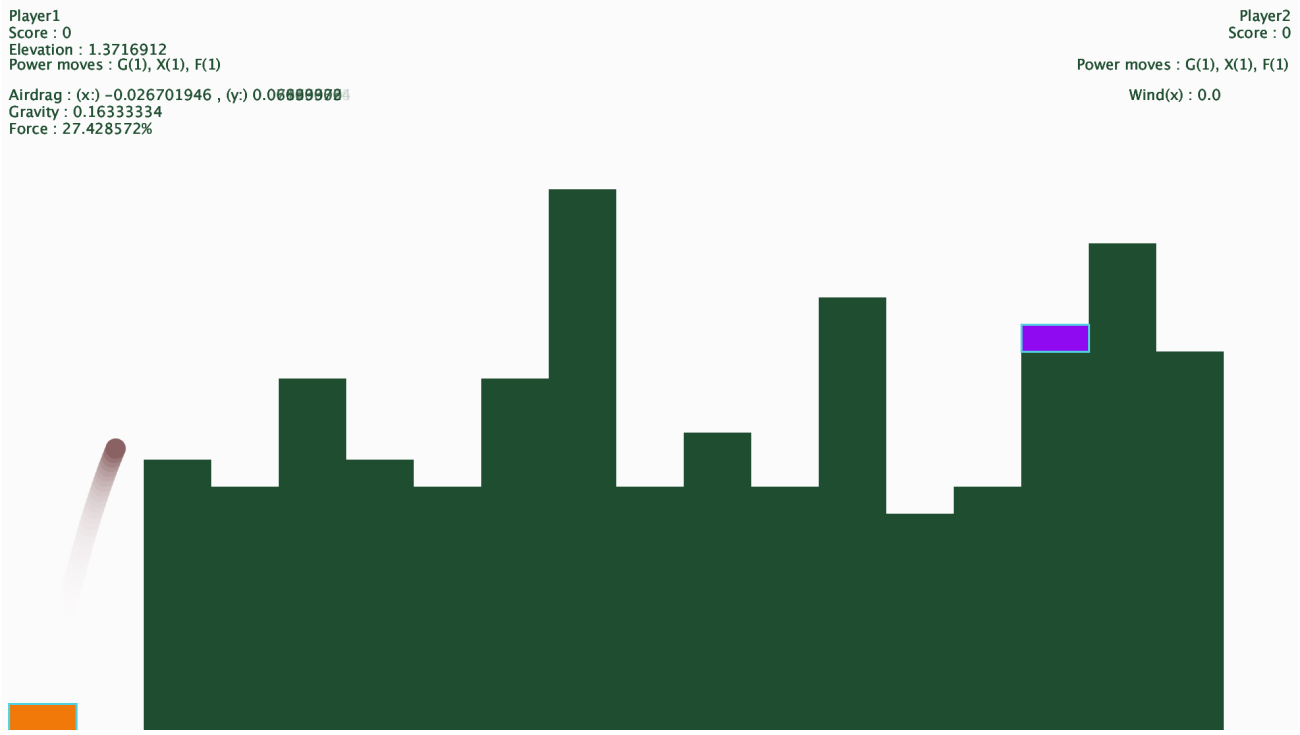
The game was developed in processing v3.5.4 .

### 2.1 Controls:

- “a” – Move Left
- “d” – Move Right
- “Mouse” – Click, aim, and drag to shoot with specific force.
  Note: Click and dragging w.r.t to the player is how the force is set.
  i.e.: clicking near the player and dragging away from the player increases the force of projectile.
- “r” – reload the whole game.
- “g” - Use gravity gun. (ability).
- “x” – Use LandSlider (ability).
- “f” – use flashBomb

## 3. Design and Implementation

The game is designed to be moderately hard for its players. The random generation of land terrain is one of the aspects that adds to this. The wind speed is another important part of the game, it could range from anywhere between .01 to 1 irrespective of its direction. The wind is locked when the player is about to fire. The player has to keep an eye on not only the previous trajectory but also at the wind speed to score. The players have various abilities which if used carefully can change the tide of the game. Some of these are implemented to make the game a little easier. While the abilities are an important part in winning the game, a seasoned player can easily win without using the same. A trail effect is given to the game for almost all of the moving objects. An example can be viewed below:




### 3.1 The Map

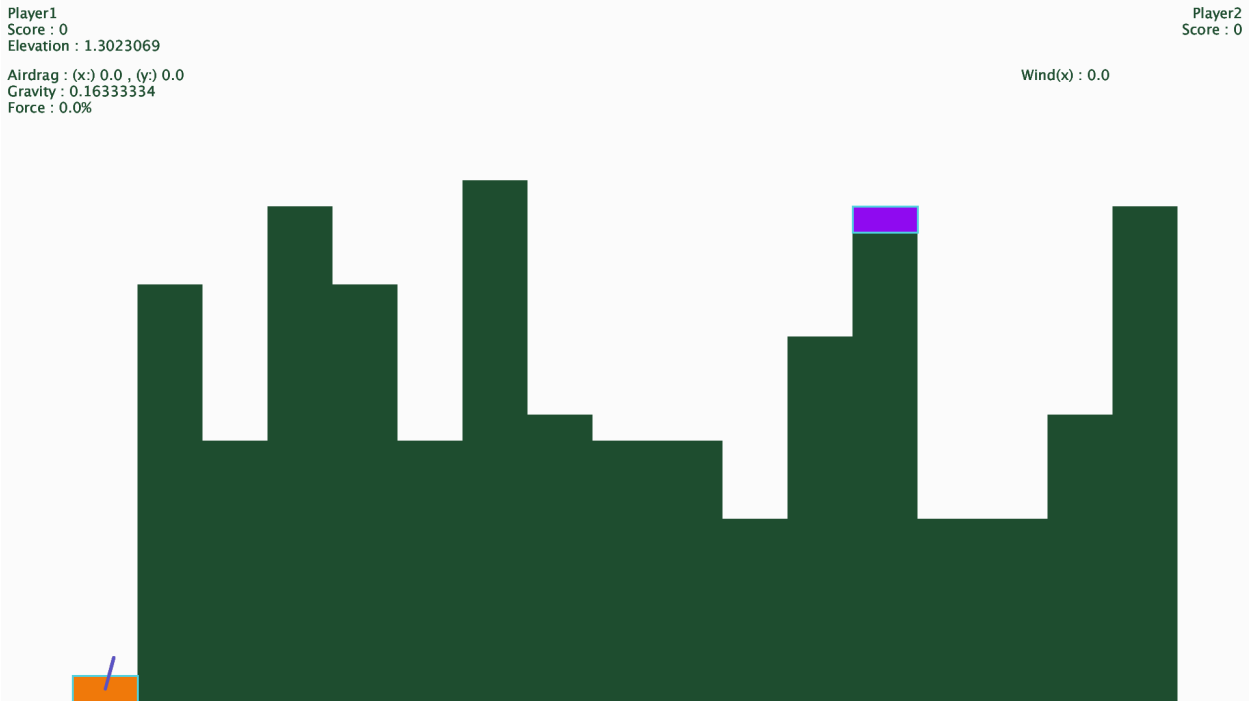
The map is made of simple destructive rectangular boxes with a specific width and height. The grid for the map is generated whenever an object of the class Map is declared and initialized with the according values. The arrayList of gridCols stores the coordinates of each of the blocks that make up the whole terrain.

#### 3.1.1 Map generation.

The whole map is generated randomly according to the space availability (width and height of the screen, and blockHeight and blockWidth given). The max number of columns are total screen width divided by blockWidth, thus giving us a how many columns can be managed. I then select a random number of columns which is neither too less nor too much. Initially I had generated only a number columns between maxCols/2 to maxCols, which made the terrain spread comparatively less than what I have implemented now. The height of the terrain specified as noOfRows is implemented in a similar manner. The height of each column is also randomly generated within a specific range of values to make the game nor too difficult neither too easy. Another important part of the map implementation was that, when I gave equal random value range for all the rows(height), I found that the height of the columns at the middle are sometimes a little too high, making the game quit tough for the players. Thus the height of the terrain at the middle section has been reduced slightly for the ease of the game.

The play area implementation was done with a screen of 1920x1080 (implemented as full Screen()) , and block sizes are 100X40. The tanks are dropped at a random location (the random value range has been picked carefully such that the tanks are not too near to each other). The tanks are dropped from the top of the screen and they fall affected by the gravity. In order to give a physical world simulation, I've made the tanks to be a little bouncy when it falls(not that the real world tanks are made of rubber, but still!!). This is taken care in updatePlayerBlocks that uses the help of

intersectsLand (to check whether the tank and land has intersected), and the velocity of the tanks to make this happen. This sometimes creates an infinite loop scenario which has been taken care by keeping number of bounces variable.
A screenshot of the final setup of terrain after implementation is provided for reference.




### 3.1.2 Forces on action

The forces applied such as gravity is low in the game when compared to real world scenario. This is due to the framerate pace. In order to create a simulation, I've used the real world gravity divided by the default framerate – 60, in order to achieve a close by of real world gravity. The max range of gravity gun is slightly higher when compared to this. This was given to make the game a little harder and more fun with greater forces in action. Even though low values for gravity can be achieved, the terrain and tanks are considered to be of great mass, that they do not float.
The wind velocity is generated randomly and then toned down by a factor of 200, to make it applicable for the game. I could go for higher values, which could make the game physics a lot harder and irritating (No one would want to fire their tanks in hurricanes!). The direction of wind is also chosen randomly with -1 to negative x direction, 1 to positive x-direction and 0 to not have wind at all. The only problem I found here was the abrupt stopping of the wind when the direction goes from -1/1 to 0. But it could be considered as very low wind velocity or drop of wind velocity that does not affect the bullet.
The drag force is dependent on the velocity as in the real world. It is not defined in a separate function but in the update Projectile function that keeps updating the bullet. Drag force here is a fraction multiple of the velocity in the opposite direction and is capped at maximum value of 1.

### 3.1.3 Extension

- Falling terrain.
  The terrain is affected by gravity(only) and the bullet. When a block of land is destroyed by

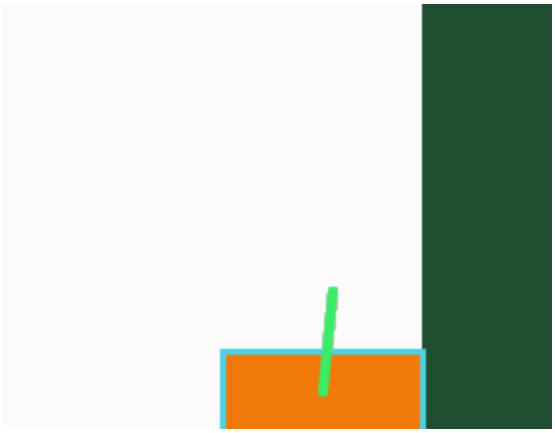
the bullet, the land above it falls down to occupy the empty space. The land do not bounce like the tanks. The gravity applied for the terrain is difference from the original gravity applied to the tanks and the projectile. So these terrains fall quite faster when compared to other objects in the game. This is given as such to achieve a sense of greater mass (If implemented as mg (force) – code complexity increases).

## 3.2 The tanks

The tanks are the differently coloured blocks in the game that falls as the game starts. To make it more realistic the tanks bounce of when they fall from a height (depends on the height) and comes to a rest due to action of gravity and a dampening factor given in updatePlayerBlocks section. A bouncing limit is also provided to make the implementation not to make the effect irritating.
A trail effect is further provided for users amusement.

### 3.2.1. Phases

There are only two phases in the game. Shooting phase and move phase. Although the move phase is not implemented specifically, whenever its not a shooting phase it's a move phase. The gamers are made known the turn by the glittering cannon of the tank that only appears at the user's turn. A tiny screenshot of the player at this stage is provided.



Explanation:

3.3 A player's turn is over when the player fires or uses one of its abilities. The player **can move** before it fires if the adjacent grid is not allocated by land. The player does not have the ability to hope on the land above. A tank can only move forward or backward with each increment being the blockWidth. The players are not allowed to leave the play area in the moving phase. The players are not limited to a specific number of moves. This was provided intentionally as the game terrain may sometimes be tough. So to ease off on the gameplay infinite movements are given.

In the **shooting phase**, the player needs to align the sparkling cannon according to direction of the fire. The start of the line is anchored to the center of the tank, as this is where the shell is fired from. However, the end of the line is anchored to the mouse. So long as the player holds down the mouse button, they are able to quickly and easily see and adjust their trajectory.The player should then click and drag the mouse to

control the force of firing, which is crucial for the gameplay. The more you drag away from the player increases the force and dragging near to the player decreases the force. The game calculates the angle to which the cannon aimed at and fires a projectile according to the force provided. Unfortunately once the player clicks for firing, the player cannot stop the tank from firing. The game mechanics is made this way to introduce difficulty to the game. Initially I was going for a gameplay mode which takes in the elevation and force from the user and then fires when the said are entered. But I felt that a visual aid goes a long way for fun gaming and toughness.

The implementation of move phases has a small glitch/bug wherein which the game does not take any keypress values at the initial stage;- i.e; when the game starts. But after both the players fire their cannon for once this glitch disappears. This is a one in a few cases scenario but according to my evaluation the bug has something to do with the bouncing effect of the tanks. The bouncing effects at times would go on an infinite bounce which I took care with the bouncing limit. Although one problem was taken care one of the side effects of the same sometimes appears (rarely).

**3.3 The shell.**

The bullet is a small ellipse with diameter 30 pixels. When fired it appears from the center of the tank and moves onward with the path according to the elevation.

The location of the projectile is kept in track with bullet Pvector. Once the projectile is fired, the force with which it is fired(only applied once), the gravity, air drag and wind controls the path of the bullet. As explained air resistance depends on the velocity of the bullet as a simulation of the real world physics. All the other forces are applied once pre frame, thus navigating the bullet accordingly. The gravity is made to be greater than air resistance and wind as in the earth. All these are then normally added to the bullet vector thus changing it per frame.

**3.3.1 Collision detection**

At each frame the position of the shell is updated and is compared with all the existing grid columns and the player tanks to detect collision. With the help of constrain function if the new location of the projectile crosses any of the boundaries of the said blocks a collision is detected. Whenever the bullet meets a map grid the specific element in the grid is destroyed(removed). But when the projectile meets either of the players, the bullet is destroyed, and the score is incremented accordingly. The bullet is destroyed(technically the vector is multiplied by zero), whenever it collides with something.

Simultaneously when the bullet is checked with the distance from each of the grid elements or the player, it is also checked whether it has left the play area. Once it leaves the play area, except for the height(upwards), the bullet is destroyed and the specific players turn comes to an end.

The implementation case of friendly fire hit, was a little complex and fun. Once the basic implementation was done the projectile was destroyed even before it left the tank, as the check case for friendly fire was completely satisfied. This was later taken care with the help of a leftTank Boolean variable which keeps tabs on whether the bullet left the tank boundaries.

Finally when any one of the player scores 5 points wins the game and a simple win screen is displayed telling which player has won the game. The game can be then reset to the start by pressing r/R key.
The final end screen is shown below:


## 3.4 Abilities

### 3.4.1 Gravity gun

A simple implementation by changing the set default gravity. When a player uses the gravity gun the gravity applied in the game is changed. The gravity is chosen randomly from between the lower and the upper value. The gravity affecting the projectile is only changed. The tank and the land is considered to be very heavy such that at lower gravities it does not float.

### 3.4.2 LandSlider

Another ability of the tanks is that it can destroy a single column of land mass that is right infront of them. I felt that this ability is crucial in the game physics as in some cases.eg: when the land in front of a player is so high that the player has difficulty in shooting.

### 3.4.3 FlashBomb.

Flashbomb is a common ability in shooting games. As the name suggests it flashes the enemy team player for a particular amount of time such that it cannot steer or navigate clearly. In this game when a player uses flashBomb, all the land terrain and the player is hidden from the enemy player. For the real world simulation of unclear navigation I have hidden the statistics details of the gameplay such as gravity, elevation etc. The effect of the flash bomb lasts till the enemy users move ends.The enemy user is only able to see himself. A screenshot of the scenario in which player 1 uses a flashbomb is shown below.

### 3.5 Game design choices

One of the implementations that makes the game a little hard is the introduction of randomness on almost every feature. Be it the land terrain generation, wind velocity, player locations or the gravity gun. The win/lose probability was drastically changed with the introduction of the same.

The introduction of abilities is another feature to talk about that introduces creativity and timing amongst the players. For example a skilled player could not even be affected by the flash bomb as the player will have a memory of the enemy location and terrain. The abilities given introduces an addition of creative gameplays.

## 4. Conclusion and evaluation

To conclude I feel that within the given time limit, I was able to create a game that simulates the real world and is also enjoyable. The game contains a map that is randomly generated with destructible blocks that fall as the supporting blocks beneath them are destroyed. There are 2 tanks that can move across the terrain without driving through blocks or outside the map, as well as being able to aim and fire a projectile with a desired power and elevation that is affected by gravity, wind and air resistance, causing it to follow a parabolic trajectory through the air. If a tank is hit by the shell, a point is awarded to the firing player. Once a player reaches a score of 5, the game ends, and the option to start a new game is given. There are abilities given to each of the player which can be used to power through the game. Basic real world physics is applied in the game such as the bouncing of tanks when falling from a height, how a projectile is affected by gravity, wind and drag.

Some of the features that could have been implemented could be the displacement of a player when a direct hit is taken. I feel that it would be important w.r.t game physics if the tank could be bounced back/forth according to the direction of the hit. Additionally another feature would be the floating land mass. When the gravity is too low(I haven’t implemented low gravity as I did not have time for this

feature), if a supporting block is removed and the blocks unsupported is less in number(say 2 blocks), they could be made to float in the game area. This low gravity concept could've introduced much more game physics and the game would've been more fun.

*Bugs/glitches*:
From personal testing I found that sometimes the first movement/ability use of players do not take place due to some bouncing glitch which I was not able to completely filter out. The glitch disappears after a round of firing.