CS4303 Video Game Report

P1: Artillery Game: Square Tank

Student Name: Hantao Sun

Student Number: 180007044

Due Date: Oct 03, 2018

1. Fundamental Structure

9) Screen

This game design is divided into different "screens", each screen with different UI are provided to players complete different operations. Screens are implemented with PGraphics class of Processing. There is a variable called *screen,* used to mark which screen should be showed now. Then the *draw()* function just need to draw and put the screen in the proper position according to the value of *screen*. Another intention is to avoid incorrectly triggering the click events of buttons. There are six screens in design now: welcome screen, instruction screen, intro-menu screen, choose-model screen, play screen and winner-tooltip screen. Design and functions of these screens would be displayed in later parts.

9) Button

Buttons are designed as a class, whose attributes includes its position, size, text, location (which screen it is put in, to call the functions of PGraphics, almost all structurers of classes who have *draw()* functions need this parameter), the position of the screen(to calculate the offset of the button). Basically, a button is a rectangle drew in a screen with some events. To make the button more pretty, the texts on the button are transferred into groups of points, with the help of Geomerative2D library. Figure 1 shows an example.

Figure 1: Introduction button in the welcome screen(normal status)

There are three status of the button: normal, mouse hovering, clicked. There are two functions（*isHovered()* and *isClicked()*） listening two unnormal status, whose return type is Boolean value, and the outlook of button is different in three status, just like showed in figure 1~3.

Figure 2: Introduction button in the welcome screen(hovered)

Figure 3: Introduction button in the welcome screen(clicked)

9) Size

The game is always full screen, and all positions of sizes of elements would change according to the size of screen. Because full-screen game could make players

more concentrate on the game, and it could enable developers design some small elements more easily(although there are not much in this game).

2. Welcome screen

Welcome screen includes three buttons and a title picture made by Photoshop, just like figure 4.

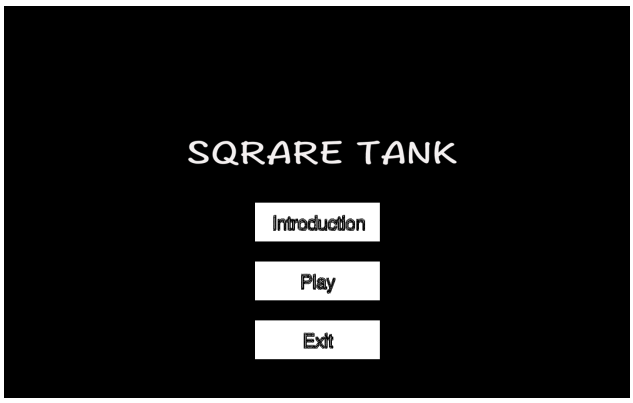

Figure 4: Welcome screen

It could be easy to understand that the "Introduction" button and the "play" button are linked with the introduction screen & intro-menu screen and the choose-model screen, while the "Exit" button is used to quit the game.

3. Introduction screen & intro-menu screen

If click the "Introduction" button in welcome screen, introduction screen and intro-menu screen would be displayed.

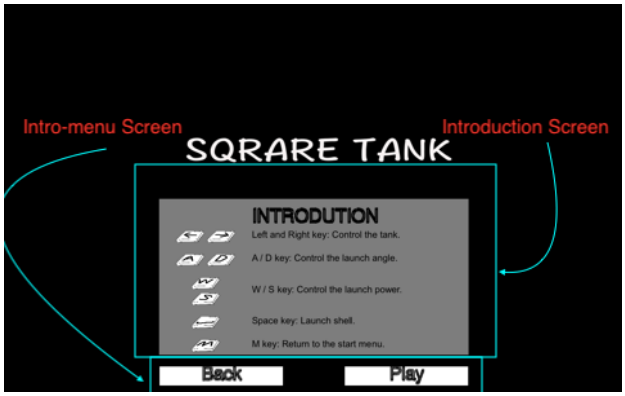

Figure 5: Introduction screen and intro-menu screen

Figure 5 indicates the layout. Two screens are not merged because putting similar elements in one image makes it more convenient to adjust position and size of elements. Introduction screen introduce some basic operations of the game. After reading it, players could choose to go back to the welcome screen or click "Play" and choose model.

4. Choose Model

There are two models provided to players: PVC (Player VS. Computer) and PVP (Player VS. Player). Player could switch the model in choose-model screen, which would be accessed by click "Play" button in welcome screen or intro-menu screen.

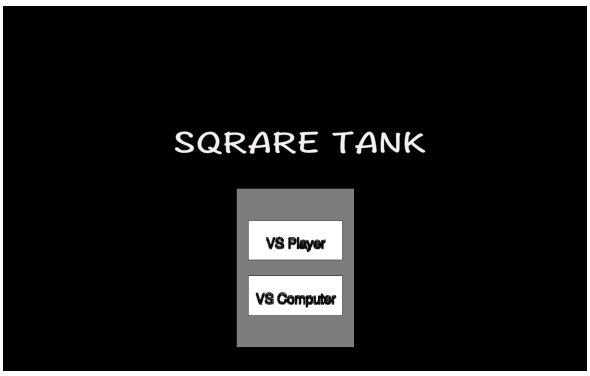

Figure 6: Choose-model screen

If player click the "Play" button in welcome screen, there would not be a introduction screen displayed. Because for those who have knew the playing method clearly, reading introduction may be meaningless and wasting time. This may be a weak point of some online e-sports games, such as *League of Legends.* Compulsive reading or tutorials could be boring for experienced players.

5. Play

This part will indicate the playing methods, design and implements of elements.

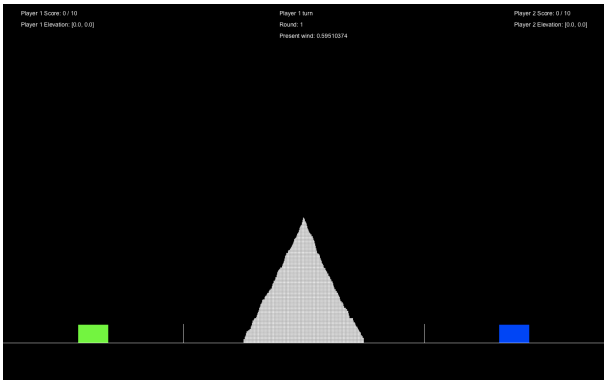

Figure 7: Playing interface

1) Basic Rules

Rules of the game match the request displayed in ***CS4303_1819_P1Intro.pdf*** .

Two tanks launch shells in turn, what players need to do is set the position of their tanks and the elevation and power of launching. If a tank is hit by a shell, its opponent could get 1 score, even if the shell comes from itself. At the end of a certain round, if there is a player has gotten 10 scores while the other has not, this player would regard as the winner. If both two players reach 10 scores in one round, the game would come to draw. The scores of two players could be checked through the information panel at the top of the screen.

There are several phases in one turn: moving, aiming, producing shell and shell flying. The current phase is marked by a variable called *next,* preventing operations which should not accessed in this phase (e.g. move tank or launch again while the shell is flying).

During playing, players could press 'M' to return to the welcome screen at anytime.

2) Play area

The available areas of two tanks are located at the most left and most right of screen respectively and have the same size. The landscape is always plain, represented by a long line. Although lumpy floor could make the game more interesting and challenging, the implement of it is quite difficult. It is also not

easy to keep balance between two players, because of the small map and the balance between floor and other elements.

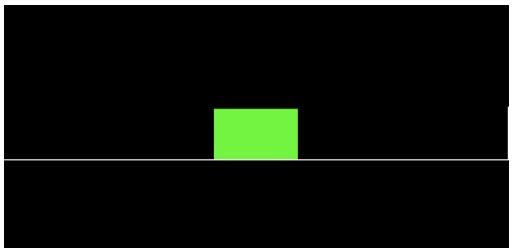

Figure 8: Play area and tank of player 1

3) Brick columns

Every time start a game there would be 200 brick columns in the mid of the floor. In codes it is represented by an array of objects of class Column. The number of bricks of each column is randomly set at the start of the game. The columns at more central positions have more bricks than those at two sides, which makes these columns always seemed like a small mountain. Otherwise, some columns may be meaningless, because it is not very possible for shells to fall into a "hole". That's also why two small gaps set between columns and play areas: to avoid building an extremely tall "mountain" and making it difficult to transcend.

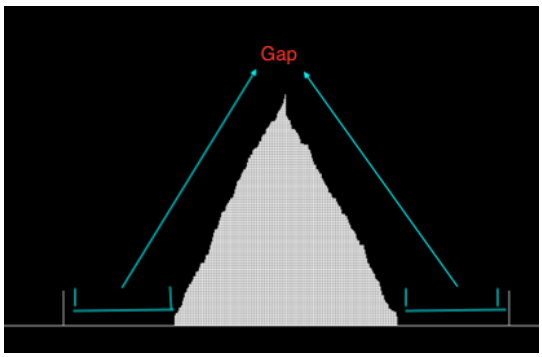

Figure 9: Brick Columns and gaps

4) Tanks

Tanks are represented by rectangles in different colours like which showed in figure 8. Using LEFT and RIGHT key could move the tank towards the left or the right, whose implement consult the codes provided in the first tutorial. In every frame, if the direction keys are pressed, the position of the tank would be recalculated. Certainly, tanks could only move in play area, if there is a frame the tank is trying to move to an illegal point (out of screen or into gap between columns and play area), the x coordinate of tank would be reset to the closest border of play area.

Moving tanks is allowed in the moving and aiming phases of a turn.

5) Aim

In moving and aiming phases, if player press A/D/W/S, the line of sight and the power gauge would appear.

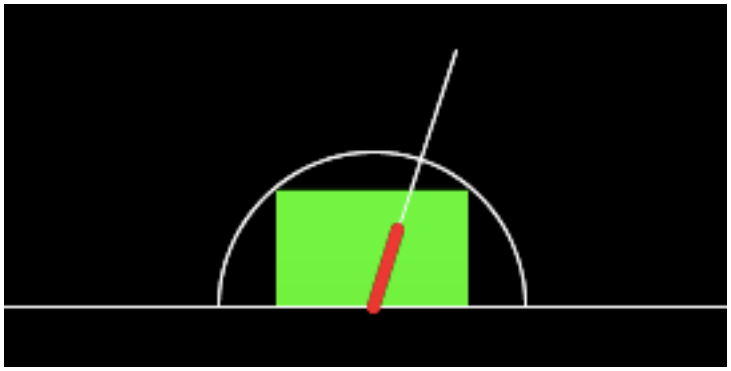

Figure 10: The line of sight and the power gauge

If player keep pressing D/A, the line of sight would rotate clockwise/anti-clockwise. While the length of red power gauge would increase/decrease if player keep pressing W/S. The former represents the elevation and angle of firing (between 0 and PI) while the latter represents the initial speed of the shell (not velocity because it is just a value but not a vector and max power is 20). At the start of each turn, the angle and power would be reset to PI/2 and 0 respectively.

This is easy to implement as long as use *translate()* function. Just translate the (0,0) to the centre of the anti-circle and rotate the value of angle, then draw the line according to the value of power. These values are supposed to change if keys discussed above are pressed. If players think it is a good time to fire, then they could press SPACE key to start to produce the shell, then they could not move tanks or adjust the angle and the power in this turn.

6) Produce shells

In producing phase, the length of red power gauge would keep reducing while the size of shell would keep increasing, making it seemed like that the shell is produced with power. This is designed because the tank is a rectangle without gun barrels. Producing animation could make the launching action not so abrupt and unreasonable. The shell is always at a same size, so that the stronger power is chosen, the process of producing would spend longer time.

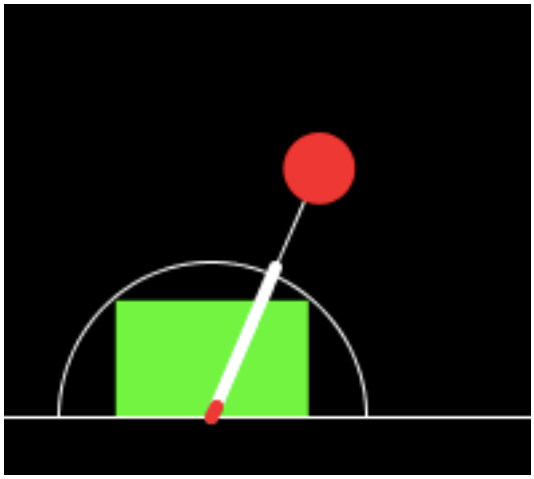

Figure 11: Tank producing shell

Once the shell is well prepared, it would be launched with an initial velocity decided by the elevation and power.

7) Movement of shells

The process of movement simulates the real movement. Objects of class Particle, the super class of class Shell, have three relevant attributes: accelerate, velocity and position, both of which are vectors. However, the difference is that the units of these attributes are (pixel/frame)/frame, pixel/frame and pixel, which is easier to understand and utilize. According to Newton' s Second Law, the acceleration is calculated by forces vectoring shells, then this acceleration would change the velocity of next frame, finally these processes would be reflected in the change of position. Classes of forces are supposed to implement functions updating forces for objects of Particle. The movement of shells is influenced by gravity and force of wind.

a) Gravity: The gravity is represented by a vector like (0, y) (y>0). The value of y

is calculated by the gravity coefficient (g), which is a vector, and the mass of particle. The expression is:

$$G = m * g$$

In fact, gravity coefficient exactly equals with the acceleration of gravity. The values of gravity coefficient and the initial velocity are coordinated (value of g is 4.9 while that of initial speed is the value of power multiply 4) so that the shell could move reasonably and its drop point could be limited in a logical range.

b) **Force of wind:** In this game, the direction of wind is always horizontal, so the force of wind could always be represented by (F, 0). The value of force is set randomly at the start time of every round to keep the fairness. This value is impossible to so large that influence the game extremely, because those who are more skilful and experienced deserve the triumph of this game, but not those who have better luck.

The velocity always decreases because of the air friction. Considering it, the velocity multiples damping coefficient in every frame. The value of damping coefficient of this game is 0.995. Just like wind force, it should not be very influential to the game.

8) Collision detection

There are two elements need to be considered about collision detection: shell and tank. The collisions of tanks and border of play area are very easy to tackle

and have been discussed, while collisions of shells are much more complex.

a) **Collisions of shells and floor:** These collisions are similar with those of tanks and borders, just compare the distance between centre point of shells and floor and the radius of shells.

b) **Collisions of shells and columns:** From a geometric point of view, these collisions are not just those of lines and shapes, which are easy to justify, but circles and rectangles, because every column could be considered as a rectangle. The solution used in this game is: find the crossover point of the circle (shell) and the line linking the centre points of both circle and the rectangle. Only if this point is actually in the rectangle, the collision of the circle and the rectangle has happened.

   If collisions of shells and columns happen, 20 bricks of the column hit by the shell and 9 columns at the left and right of this one would disappear. Because if there are just few bricks (e.g. only one) destroyed, this design would not influence the strategy of players, making it meaningless. However, it seems silly and unreasonable if too many bricks suddenly disappear without an explosion effects, which is so time-consuming and a little difficult.

Figure 12: Columns losing bricks

c) **Collisions of shells and tanks:** The detection method is quite similar with that of collisions of shells and columns. If the shell collides a tank, the player controlling the other tank would get 1 point.

9) AI

In PVC model, the tank of player 2 is controlled by a simple AI. In the first round, the position of tank, the angle and power of launching are random in a suitable range (e.g. the position of tank would be the point axisymmetric with tank of player 1). Then AI would adjust these parameters according to the results of firing of both player 1 and itself.

a) **If player 1 move:** Normally AI would change the power, but if the power is large or small enough, it would adjust the angle.

b) **After AI fires:** Just like the first situation, if the drop point is at the left/right of player 1, the AI would choose to decrease/increase the power or the angle, according to the current power.

c) **After player 1 fires:** The principal is to flee the drop point of player 1' s

shell from current position, and the angle would be adjusted automatically based on the change of the position of the tank. For instance, if the shell fall left the AI tank, in AI' s turn, the tank would move towards the right and the angel would increase to compensate the larger distance between two tanks caused by this change (not to adjust power because the compensation should be limited in a rational range). However, if the tank is currently at the corner of play area, the tank would change the direction of fleeing, in case of it keep staying at the corner for rounds and waiting for the shell. If the AI tank is hit, it would flee towards left or right randomly.

6. Winner tooltip

At the end of a round, if there are at least one player have gotten 10 scores, the winner-tooltip screen would appear automatically.

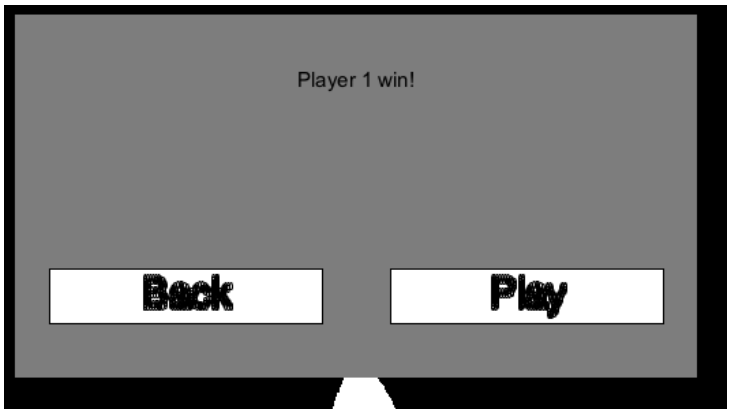

Figure 13: Winner tooltip

Players could choose to click "Back" button to return to the welcome screen

or click “Play” button to restart a game in the same model, because most of players would keep playing alone or with another player.

7. Extensions

There is something in design not done because of the limitation of time and could be extended in later time.

1) Items with extra score, affection of gravity on bricks, sounds (all mentioned in ***CS4303_1819_P1Intro.pdf)***
2) Items with one-off buffs, e.g. 1. In next round, you would get more scores if you hit the other tank; 2. In next round, you could move very slowly to avoid the shell.
3) Different kinds of shells and weather system, e.g. Napalm bomb could persistently get 1 score in every 5 rounds if it hit the target, which would be buffed by windy weather, a weather has more powerful wind, and get DEBUFF from rain, a weather with weak wind and larger air friction.
4) Achievement system, just like the trophy system of PlayStation.
5) User experience optimization, e.g. A confirmation tooltip if players press ‘M’ to return to the welcome screen, in case of maloperation.