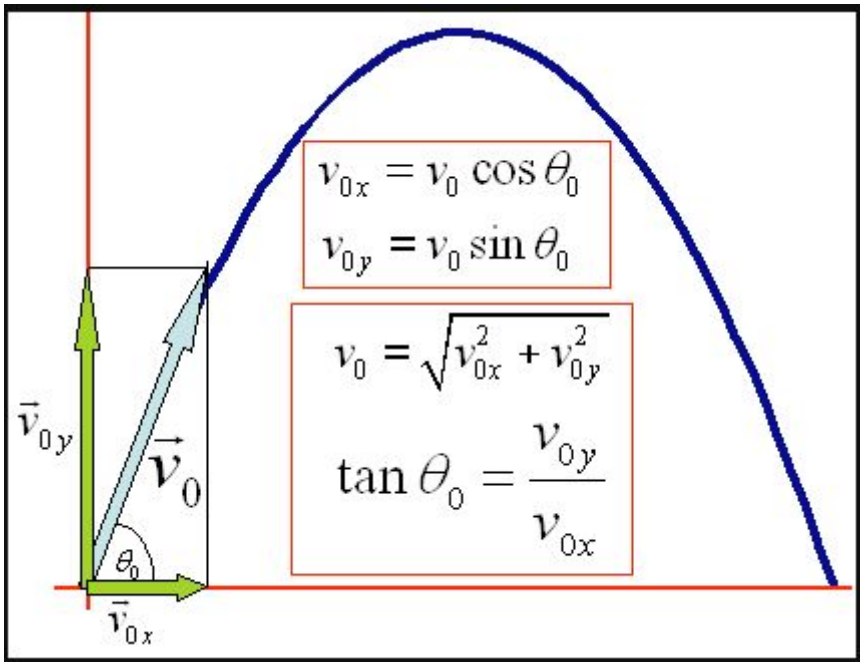
Preliminary SDD Assessment Task 4 - Group Task

Defining and Understanding

Statement of intent:
We intend to simulate the natural phenomenon of projectile motion through the creation of a tank shooting game. Projectile motion will be present through the curve of the tank shot.

Research:
In perfect conditions without wind resistance or any other factors, projectiles will follow a parabolic curve. The variables necessary to determine this curve, are the starting position, starting angle and starting velocity. We will use these variables and mathematical formulas to calculate the curve of a parabola.

Formula:



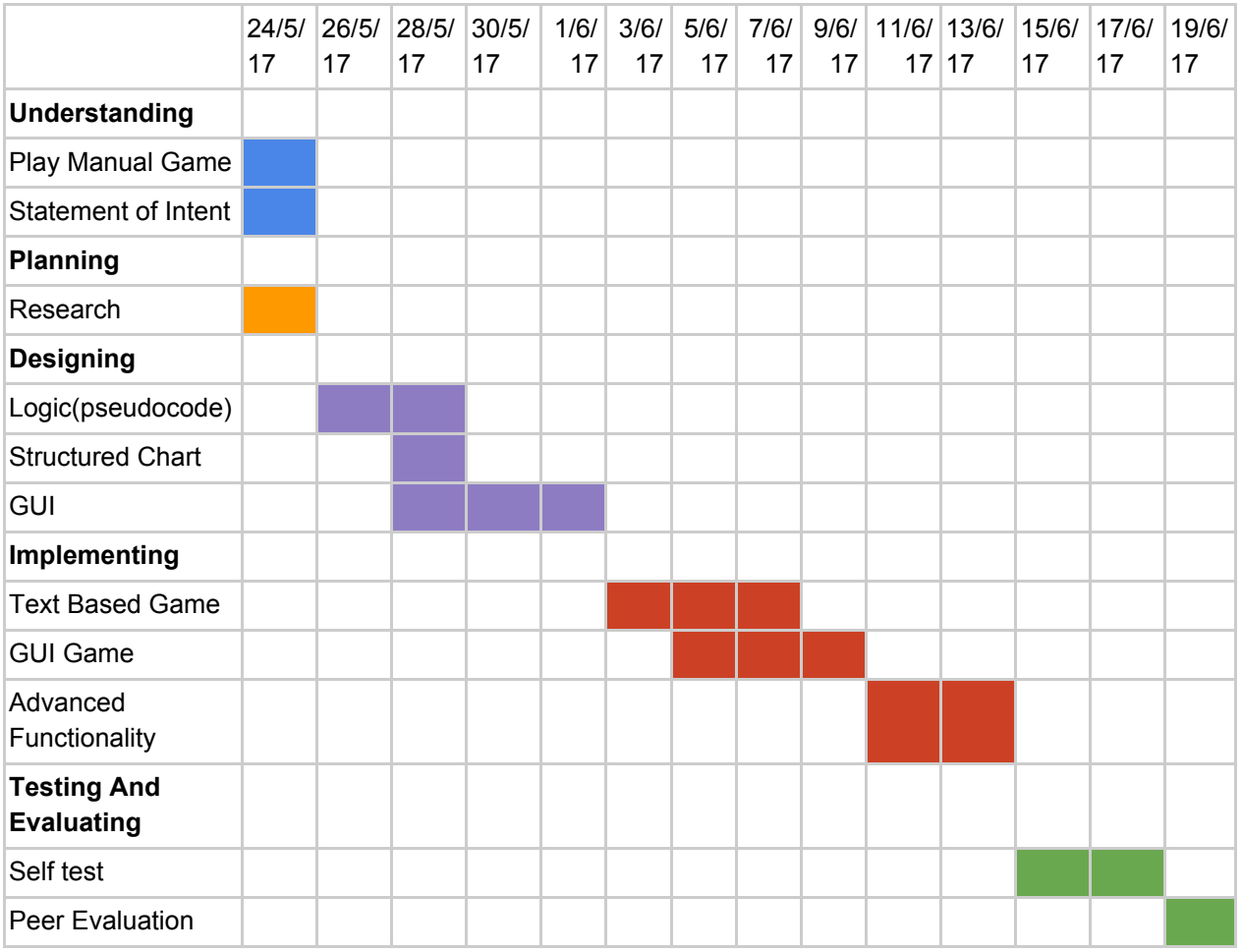
As displayed by the above picture, trigonometry is required in order to determine x and y coordinates of the curve followed by the projectile. Assuming the initial angle of launch as O. Cos(O) is used to determine horizontal movement and Sin(O) to determine the vertical movement. Both are these are multiplied by time to update the position over the entire air time. x coordinates is a linear increase, whereas y coordinates follows a parabolic pattern.
The entire parabola will have to have a vertex, that is the maximum vertical point where positive gradient turns to negative gradient. This will be influenced by minusing by

gravity * times^2. Following this, y will keep increasing until gravity * times^2 becomes greater than it.

Planning

Development Approach:
A prototyping approach will be used to create this simulation as while we have a general idea of how our program is to function, we do not have the specifics down. By continuously prototyping and testing, we can have continuous feedback on our project to know what to improve on. This way we can slowly reach a much more full functioned simulation, instead of one strictly following our original ideas.

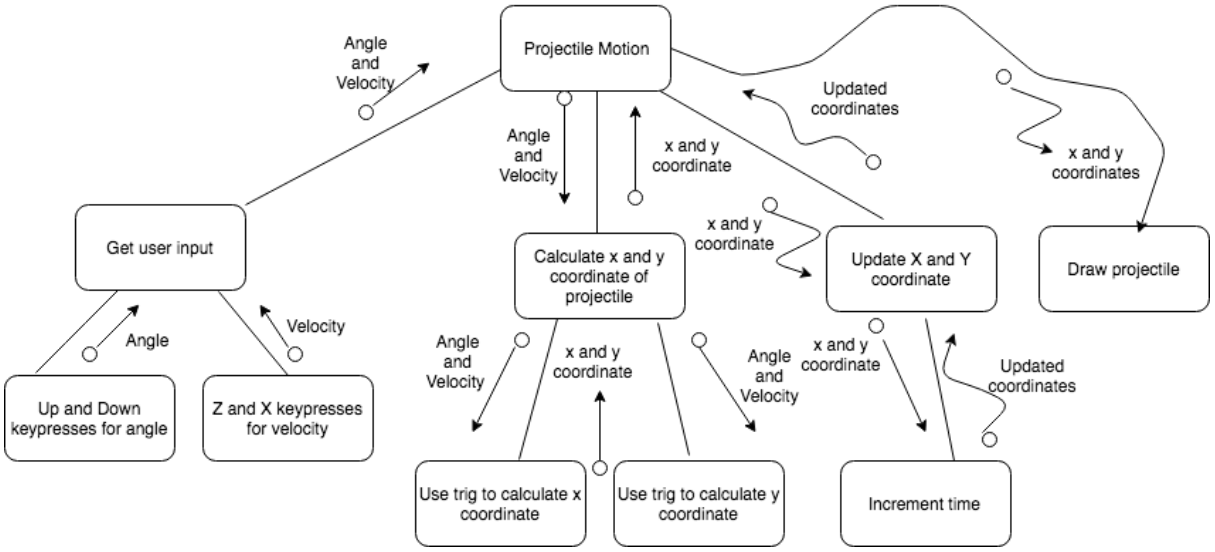
Gantt Chart:



IPO Chart:

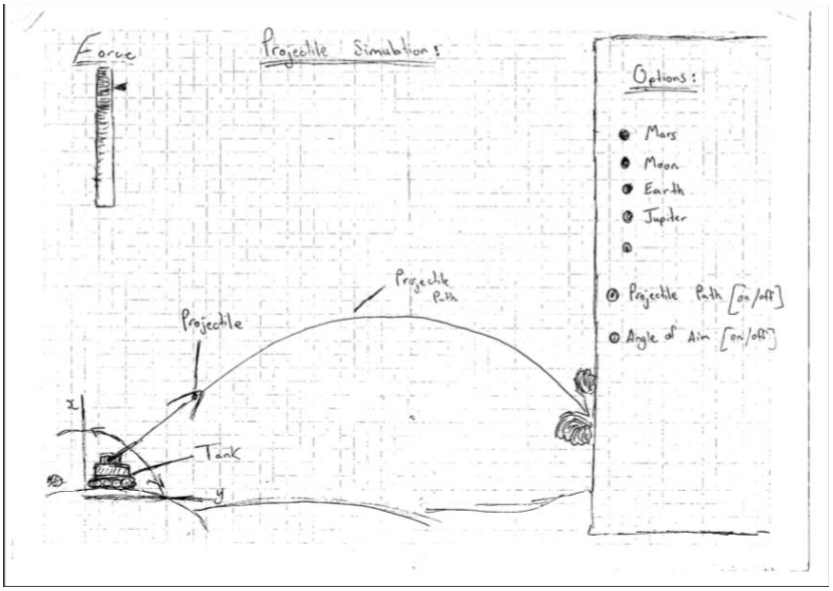
Input	Process	Output
Angle, velocity and starting position of projectile	Perform the necessary mathematical calculations	Updated X and Y coordinate of projectile
X and Y coordinates	Draw projectile to coordinates and increment time to calculations	Moving projectile

Structure Chart:



Designing

Screen Design:



Pseudocode (3 main algorithms):

```
BEGIN

VAR ANGLE = 0
VAR VELOCITY = 0
VAR TIME = 0
GRAVITY = 0

FUNCTION ANGLE():
        IF UP.KEYPRESS = TRUE:
                ANGLE += 1
        IF DOWN.KEYPRESS = TRUE:
                ANGLE -= 1
FUNCTION VELOCITY():
        IF Z.KEYPRESS = TRUE:
                VELOCITY += 1
        IF X.KEYPRESS = TRUE:
                VELOCITY -= 1
FUNCTION PROJECTILE():
        PROJECTILE.X = START + VELOCITY * COS(ANGLE) * TIME
        PROJECTILE.Y = VELOCITY * SIN(ANGLE) * TIME - (GRAVITY*TIME^2)

        DRAW
        ARC(PROJECTILE.X, PROJECTILE.Y, XRadius)
        ENDDRAW

        TIME = TIME += 1
END
```

Implementing

Logbook:

Date	Who is Designing / Programming	What is planned to be achieved	Successful or Not	Issues	How Issues were solved
7/8	Vincent starts physics research and finds formulas for trajectories Dejan finished Gantt chart, planned/drafted designs for simulation.	Finish drafts	Successful		

10/8	Vincent attempts to code the projectile movement within javascript. Dejan started structure chart in Photoshop.	To have the projectile working properly in javascript. Finish charts including IPO and Structure charts.	Unsuccessful	Inaccurate trajectory that doesn't follow a proper curve.	Used a reference from http://www.cs.mtu.edu/~shene/COURSES/cs201/NOTES/chap02/projectile.html
11/8	Vincent uses above references and does further research on the maths required such as trig. Dejan designed and storyboarded various designs such as the overall GUI as well as creating the first tank image	Attempt to code working projectile Design our GUI	Successful		
12/8	Vincent implements options for user keyboard input in the simulation.	To code a key listener for the user to hit spacebar in order to shoot the projectile. Allow users keyboard input to modify initial angle and velocity.	Unsuccessful	Program not correctly receiving space bar presses. Angle could be modified while previous projectile was still in motion, causing said projectile to change trajectories in mid air.	Caused by incorrect variable names and misplacement of if statements. Corrected variable names and placed if statements inside move Projectile function. Only allow the changing of angle while the projectile is off screen.

13/8	Dejan creates more images for the tanks projectile angle	Have all tank images resized and ready for code.	Successful		
16/8	Dejan finishes tank designs while Vincent implements them into Javascript	To input all finished tank images into javascript in working order.	Successful		
20/8	Vincent implements targets into game	Implement various targets	Unsuccessful	Target not detecting collision.	
25/8	Vincent implements hit detection and further stylise GUI	Implement working hit detection Style background and create range sliders Write in necessary information such as angle, velocity and instructions	Unsuccessful	Score updating even when target not hit by projectile	Range was actually inaccurate as it was actually concatenating the y and height values of the target instead of adding them. Added + before variables to force them to be numbers

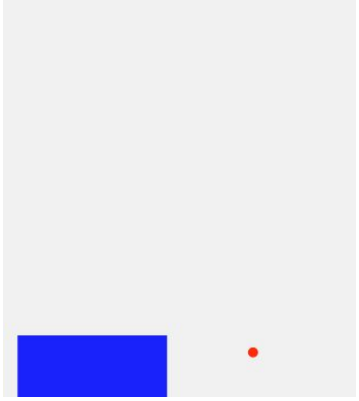
Logbook images:

11/8

Implementing physics formulas to define drawing curve

```
function drawProjectile(){
    // work out the new position of the ball
    x = projectile.x + projectile.speed * Math.cos(projectile.angle * Math.PI/180) * projectile.time ; //Velocity * cos(angle) * time + initialX
    y = projectile.speed * Math.sin(projectile.angle * Math.PI/180) * projectile.time - (projectile.gravity * Math.pow(projectile.time,2) / 2); //Velocity * sin(ballangle) * (
    time) - (gravity * time^2)
    y = projectile.y - y; // y = iy - y: initial y - above calculated y (so the shot actually starts from the tank)

    // draw the ball to the above determined ositions
    ctx.beginPath();
    ctx.arc(x, y, 5, 0, Math.PI*2);
    ctx.fillStyle = "#ff0000";
    ctx.fill();
    ctx.closePath();
```



13/8

Keyboard input

```
function keyDownHandler(e) {
    if(e.keyCode == 39) {
        rightPressed = true;
    }
    if(e.keyCode == 37) {
        leftPressed = true;
    }
    if(e.keyCode == 32){
        spacePressed = true;
    }
    if(e.keyCode == 38){
        upPressed = true;
    }
    if(e.keyCode == 40){
        downPressed = true;
    }
    if(e.keyCode == 90){
        zPressed = true;
    }
     if(e.keyCode == 88){
        xPressed = true;
    }

}
function keyUpHandler(e) {
    if(e.keyCode == 39) {
        rightPressed = false;
    }
    if(e.keyCode == 37) {
        leftPressed = false;
    }
    if(e.keyCode == 32){
        spacePressed = false;
    }
    if(e.keyCode == 38){
        upPressed = false;
    }
    if(e.keyCode == 40){
        downPressed = false;
    }
    if(e.keyCode == 90){
        zPressed = false;
    }
    if(e.keyCode == 88){
        xPressed = false;
    }
}
```

```
if(rightPressed && tankX < canvas.width-tankWidth) {
    tankX += 1;
}
if(leftPressed && tankX> 0) {
    tankX -= 1;
}
```

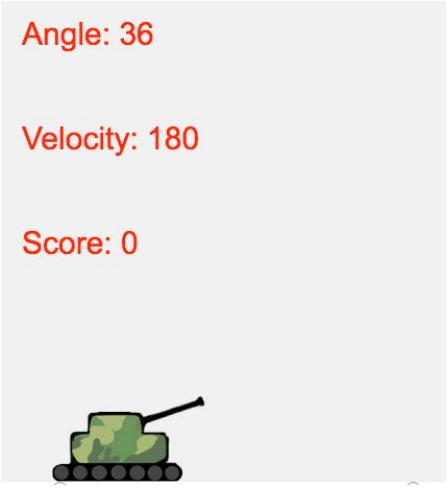
16/8:

Tank design

Implementation into Javascript

```
function drawTank() {
    if(projectile.angle >= 0 && projectile.angle <= 14){
        ctx.drawImage(image1,tankX,canvas.height-tankHeight);
}
    if(projectile.angle >= 15 && projectile.angle <= 29){
        ctx.drawImage(image2,tankX,canvas.height-tankHeight-4);
    }

    if(projectile.angle >= 30 && projectile.angle <= 44){
        ctx.drawImage(image3,tankX,canvas.height-tankHeight-16);
    }
    if(projectile.angle >= 45 && projectile.angle <= 59){
        ctx.drawImage(image4,tankX-9,canvas.height-tankHeight-52);
    }
    if(projectile.angle >= 60 && projectile.angle <= 74){
        ctx.drawImage(image5,tankX,canvas.height-tankHeight-53);
    }
    if(projectile.angle >= 75 && projectile.angle <=90){
        ctx.drawImage(image6,tankX,canvas.height-tankHeight-61);
    }
```



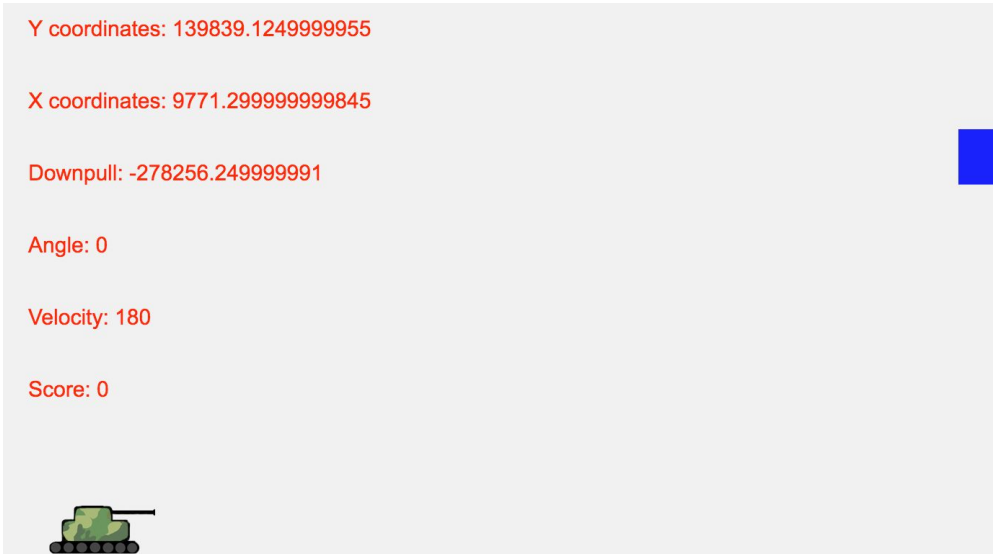
20/8

Implementing target

```
function drawTarget(){
    ctx.beginPath();
    ctx.rect(target.x, target.y, 50, target.height);
    ctx.fillStyle = "#0000FF";
    ctx.fill();
    ctx.closePath();
    target.y = target.y + target.dy;

    if(target.y + target.dy > canvasheight - target.height){
        target.dy = -target.dy
    }
    if(target.y + target.dy < 0){
        target.dy = -target.dy
    }

}
```



23/8

Background

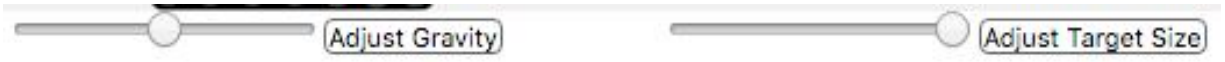
25/8

Range sliders:

```
<div id="button1">
<input type="range" id="gravityRange" value="150" max="300" min ="0">
<button onclick="gravityadjust()">Adjust Gravity</button>
<p id="gravityadjust"></p>
</div>
```

```
function gravityadjust() {
    var gravityadjust = document.getElementById("gravityRange").value;
    document.getElementById("gravityadjust").innerHTML = gravityadjust;
    projectile.gravity = gravityadjust;

}
```



Finished stylised game:

Testing Documentation:

Input	Expected Output	Actual Output	Solution
Space press	Projectile to shoot given previous projectile is offscreen	Expected output	
Up arrow press	Angle to increase given previous projectile is offscreen	Expected output	
Right arrow press for 7 seconds	Tank to move right continuously and reach roughly halfway across the screen	Roughly accurate to expected output	
Range slider pull to the left, then “Adjust Gravity” button pressed	Gravity set to 0. Projectiles should float.	Expected output	
Range slider pull to the left, then “Adjust Target Speed” button pressed	Target should stop moving	Target disappears	Removed as it would be too difficult to implement
Holding up arrow	Tank image should	Expected output	

and increasing angle by 15 degrees	update to show the angle change		
Holding up arrow to increase angle at 90 degrees	Angle should not increase any further as it is limited to <= 90	Expected output	
Space press at 90 degrees	Projectile should go straight up then down. Y value should increase then decrease, X value should not change	X coordinate output shows minor changes. This is not significant enough and is possibly due to rounding of numbers	
Space press at 45 degrees	This should achieve the maximum horizontal distance of the projectile due to how right triangles work	Expected output	
x,y coordinates of the ball equal to x,y coordinates of target	This should count as a collision, and update the score	Problems with either updating score when not supposed to or not updating score at all	Allow x to be a constant as the target doesn't move left to right. And set the range for y that counts as a collision.
Projectile fitting into range of x and y coordinates	This should count as a collision and update score	Score updates even when projectile is outside range	Range was actually inaccurate as it was actually concatenating the y and height values of the target instead of adding them. Added + before variables to force them to be numbers

Evaluation

The game / simulation we created has come to a satisfactory level of completion. It has features we designed and intended it to have from the beginning of our project. In terms of graphics, the Tank created looks good, however the background and other objects could look nicer given more time. The GUI is user friendly and we believe it is very straightforward to use. The most important aspect of the drawing of the projectile curve, we believe is done well as it matches what would be expected to happen in real life given perfect conditions. If

given more time to research however, we would have liked to implemented extra variables such as wind resistance or different projectiles that can have different travels.

Personal Contribution:

Dejan (40%):

Created and designed the main look for the game including the tank images and canvas look. Helped assist coding some aspects of the game and cleaning up code. Planned and designed on how the game would look and function as well as completing most of the documentation.

Vincent (60%):

Came up with the idea of the projectile simulation/game. Did the the Physics related research and found the necessary trig formulas. Coded the main functionality of the projectile curve as well as taking user input for velocity and angle. Coded movement of tank and moving target.

Audience Feedback:

Thomas: The game could be made more interesting by progressively increasing difficulty. For each hit, either make the target travel faster or the target smaller. Also exponentially increasing score, for continuous combo hits with no misses could be an interesting idea. But overall good game.

Mopy: Nice game, but it would have been nice to have created more tank angles, as the projectile coming out doesn't always match the tank turret. Also there are issues on my device where moving tank causes page to scroll. Would be better if everything scales to screen size to counter these scrolling issues.

Elo: Nice simulation, but simulation could be more interesting by implementing more variables such as wind resistance and different projectiles.