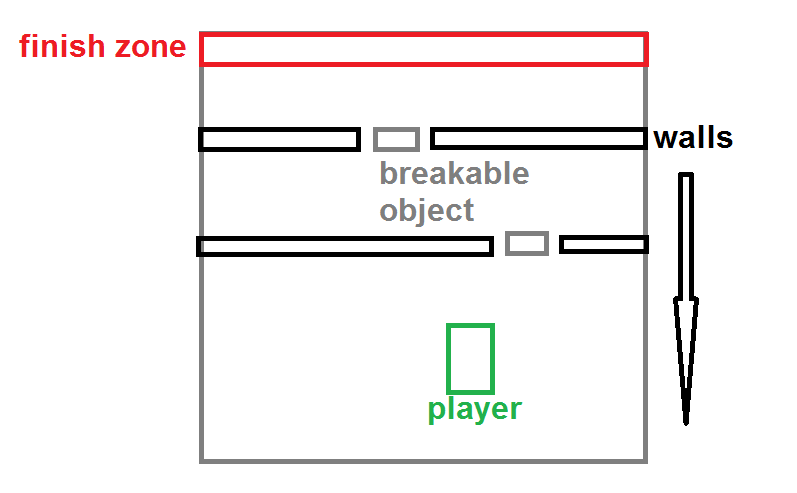
Adam Chell N0815742

SOFT10111 Javascript Game Report

**Game Initial Idea:**

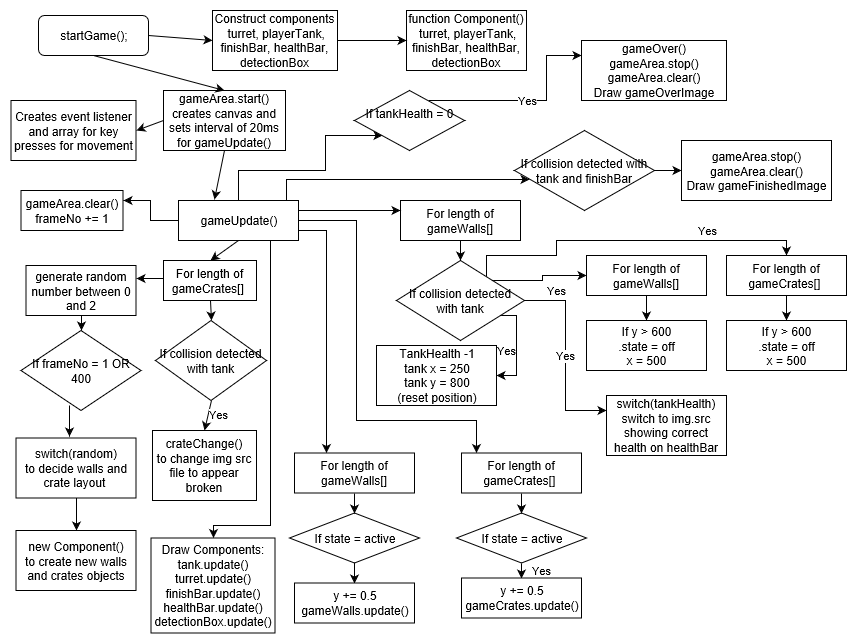
The initial idea for my game was to have a tank on the screen which was controlled by the player’s input. This tank should be able to move in all directions by using some form of rotation and being able to pivot. The tank would have to avoid obstacles such as walls that come down the screen vertically by moving it through the gaps. The gaps would have an obstacle in that can actually be removed upon colliding into it with the player’s tank. However, if the player was to hit the obstacle wall, they would then die and have to start from the beginning. The aim of the game is to get the tank to the top of the screen where the user would be able to finish the game.

**Specification / Analysis of Requirements of Program**

The program must be able to:

* Create a canvas element within the HTML webpage which will be used by the JavaScript to draw and transform all of the components which make up the game. The canvas provides a coordinate system which will be used for movement and rotation between the different images and shapes on screen. This is necessary as creating shapes and animation using html only is not supported.
* Display many different components of the game on the screen at once and update their positions all at the same time at a set interval. This will be achieved by clearing the canvas and redrawing each component at the set interval so that they appear to be animated and move. This is so that the game appears smooth and changes are made instantly to the screen.
* Detect collisions between the player’s tank and other obstacles on the screen such as the removable objects and walls by using the canvas’ coordinate system to compare different values of each component to see if they overlap. Depending on the components overlapping, different code will be carried out as a result.
* Draw obstacle walls to the screen automatically at a set interval. These walls must be able to move vertically down the screen towards the player’s tank at a set speed. The layout of these walls must also be randomised so that the gap between the walls is not always in the same position, as this would make it too easy for the player to guess where they need to move next.
* Allow the tank to be controlled by the user’s keyboard input, using the up and down arrows to accelerate and reverse and the left and right arrows to pivot and change the angle at which the tank is facing. This therefore allows the tank to appear more realistic and satisfying to control and allows for 360 movement.
* Allow the tank to be controlled by multiple input keys at the same time, such as the up and right arrow key. This will mean that the tank will be able to move on a curved line and be able to move diagonally. This will be achieved by creating an array of active keys being pressed and functions then being carried out from these.
* Show the player’s tank’s current health at the bottom of the canvas with the use of graphics. This will be done by linking a collision with -1 from the tank’s health and therefore changing the amount of health shown on screen.
* Allow the user to start and restart the game. This can either be part way through the current game or at the Game Over screen if the player dies. This will be achieved by using a button on the HTML page that refreshes the page and therefore reloads the canvas and JavaScript behind it.
* Clear the bottom of the canvas when a life is lost, so that the player will start from the bottom of the screen and the clearing will therefore clear the bottom third to avoid instantly being hit by a wall upon respawning.
* Display a Game Over and a Well Done screen depending upon the outcome of the last game played. This will be done by creating two functions that clear the canvas and present a new drawing to the user to inform them on if they won or not, which are called up if the player’s tank loses all of its lives or when the tank collides with the finish zone.

**Pseudo Code:***This program uses a component constructor and intervals to create the different objects and then update them (clear canvas, re-draw to canvas) multiple times per second to achieve animated effect.*Declare tank health variable as 3  
Create list to store game wall objects   
Create list to store game breakable objects  
Create game canvas object and assign dimensions  
Run event listener to store key presses to list  
Set interval for updating game canvas  
Set interval for generating new obstacles  
User presses start button  
Create new component for player tank and assign properties  
Create new component for player detection box   
Create new component for finish zone  
Create new component for health bar  
While the update canvas interval is running:  
 - Clear the game canvas   
 - Generate random number  
 - Use random number to assign obstacle layout  
 - Generate all obstacles in layout on interval   
 -If up key is pressed increase tank speed  
 -If down key is pressed decrease tank speed  
 -If left key is pressed change tank angle  
 - If right key is pressed change tank angle  
 - Calculate new tank position using X += speed \* angle  
 - Calculate new detect box position using tanks coords  
 - For all values in the walls list  
 - Check collision between wall and detect box  
 -If wall has collided with detect box  
 - Return collide is true and tank health -1  
 - Move tank coordinates to default position  
 - Update health bar to reflect -1 health  
 - If walls have coordinates in lower third remove those walls  
 - If breakable objects have coordinates in lower third remove them  
 -Check collision between breakable object and detect box  
 -If breakable object has collided with detect box  
 -breakable object image source changed to represent broken  
 -Check collision between finish bar and detect box  
 - If finish bar and detect box have collided  
 - Stop game canvas interval  
 - Display Well Done screen   
 - If user presses restart button  
 - Refresh webpage  
 - Draw player tank to canvas using new position created  
 - Draw obstacle positions using y cords – speed for vertical movement  
 - If tank health = 0  
 - Stop game canvas interval  
 - Display Game Over screen  
  
**Flow Chart:**



Yes

Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Description | Type of Data | Data Input | Expected Outcome | Actual  Outcome |
| Moving the tank | Normal | Up Arrow Key | The tank should move vertically up the canvas. | The tank moved vertically up the canvas. |
| Moving the tank | Extreme | Up + Right Arrow Keys | The tank should start to move in a circular motion going clockwise. | The tank moved in a circular motion going clockwise. |
| Moving the Tank | Erroneous | Up + Down Arrow Keys | The tank should stay stationary as the speed should cancel out. | The tank stayed stationary and didn’t move as the two keys cancelled one another out. |
| Colliding the tank with a Wall | Normal | Up Arrow Key towards a wall | The tank should hit the wall and then be moved back into the start position and have one less heart on the healthBar. | The tank moved towards the wall up the canvas and as soon as it touched the wall it was moved back to the start position and the healthBar updated displaying 2 hearts. |
| Colliding the tank with a Wall | Extreme | Up Arrow Key towards a wall inside edge | The tank should hit the wall’s inside edge and then be moved back into the start position and have one less heart on the healthBar. | The tank moved towards the wall up the canvas and as soon as it touched the wall it was moved back to the start position and the healthBar updated displaying 2 hearts. |
| Colliding the tank with a Wall | Erroneous | Up Arrow Key towards a crate between two walls. | The tank should hit the crate and the crate should disappear and the tank should be unaffected. | The tank moved towards the crates between the wall and it hit the crate which changed its image to the broken crate and carried on moving forward. |
| Moving the tank to the edge of the page | Erroneous | Use the right or left arrow key + forward to go left or right on the canvas. | The tank should hit the edge of the canvas and stop. | The tank carried on and went outside of the canvas width. `` |
| Colliding the tank with the finishBar | Normal | Use the arrow keys to guide the tank towards the finish. | The tank should hit the finishBar and the game should stop and the well done screen should appear. | The tank hit the finishBar and the game got stopped and the Well Done screen was loaded to the canvas. |
| Losing 3 hearts in a single game. | Normal | Use the arrow keys to guide the tank around the canvas, hitting the walls 3 times. | The game should stop and the Game Over screen should be drawn to the canvas. | Upon hitting the 3rd wall, the game stopped and the Game Over screen was drawn to the canvas. |
| Clicking the Start Game button during the game. | Erroneous | Click the Start Game button. | The button should be disabled while the game is in play and only be enabled once the game is over or restarted. | The start button was disabled after pressing it initially to start the game and only became enabled after pressing the Restart Button. |

**Evaluation:**

**What went well:**

I feel that the program I produced met nearly all my specification points that I set out to meet at the beginning of this project which I am pleased about. All the core functions are there and working and the game is very much playable. I was most impressed with how the tank managed to move on screen as it was very smooth and effective. I thought that the turret being on a separate component to the tank body worked well as it gave a realistic effect. One of the best pieces of code in this program is the function that generates a random variable which is then assigned to an obstacle layout within a switch function. This worked well as it meant that it was hard for the player to predict where the crates would be next, which made the game more interesting.

One of the more satisfying parts to this program which I have not written before is the use of drawImage to draw images onto the canvas which can then be transformed with its coordinates. The use of coordinates within canvases is enjoyable as I do not come across this often and it provides an interesting way of looking at movement and rotation. I had never used canvas before this, so it was beneficial to me learning this code. Another new piece of code that I made excellent use of is the switch statement, which was used as a quicker way of doing multiple simple if statements.

**Things I would improve if I were to do it again:**

I would like to attempt to add enemy players which could appear at a set interval in which they would shoot rounds at the player’s tank and the player would have to move to dodge them. This then brings in the idea of being able to fire rounds from the tank instead of just colliding and destroying the obstacles. I did attempt this initially however I wanted the rounds to fire in the exact direction the tank was facing, and this proved to be difficult especially as it was my first-time using canvas.

I would also like to add more buttons for the player to click at the beginning of the game which would allow them to change various variables such as difficulty, tank health, wall speed, intervals for generating. This would not too hard to implement however I felt I focused too much on the technical side of the game, and therefore lost focus on some of the simpler aspects. Another thing I would like to include is customisation of the tank object such as the appearance, or weapons, which could be upgraded with points earned within a point system for completing levels / destroying crates / enemies.

I felt it was hard to come up with a detailed and planned idea for this project as I hadn’t covered canvas elements before this in the lectures or labs. Some of the material that I needed was only explained later in the lectures when the code was already written which meant I had to keep going back to update specific parts. I wrote all the code myself and did not copy and paste any section of it, however I did use developer.mozilla.org and w3schools for help with the constructing of components and helping me understand the collision detection better. However, I feel this is justified as since canvas uses coordinates, it is inevitable that the function for collision will be similar as it is performing the same calculation, so I felt it would be pointless to try to come up with a completely new arrangement.