Tank Escape

Ryley Goodine, Zheyuan Xu

# Introduction

The goal of the game is for the player to move the blue cube tank to the end of the maze without touching the enemy cube tank.

Challenges faced with implementing the game were in the creation of the maze, how the player will be colliding with the maze walls, or the enemy cube tank and in initializing projectiles.

~~Challenges where faced with deciding on the size of the maze and its complexity, as the objects defining each wall had their own texture~~

~~Movement of the player is done through a standard browser game mapped to the WASD key arrangement. Rotation of the first-person perspective camera is given to the Q and E key and rotates left and right respectively. Changing the camera is done through the F key, which cycles through a top-down view and a first-person perspective view.~~

# Methods

## Playing Field / Terrain

* Summary of functionality
  + Plane object for floor and cubes for walls.
  + All made using Zacks game engine, with texture and colour added
* Link to theory
* Implementation details

## Main Player

* Summary of functionality
  + The player object is a basic cube shape with colour calculated using the Bling-Phong model and diffuse texturing of the alien.jpg image.
  + The player is moved using the standard browser WASD key configuration.

this.player.translate(vec3.fromValues(+0.x, +0.0, +0.0)); // move left

this.player.translate(vec3.fromValues(-0.x, +0.0, +0.0)); // move right

this.player.translate(vec3.fromValues(+0.0, +0.0, +0.z)); // move forward

this.player.translate(vec3.fromValues(+0.0, +0.0, -0.z)); // move back

* Link to theory:
  + Movement of the player object on the scene is done through constant translation of the player objects current location. The movement in the x-axis is considered flipped because
* Implementation details
  + This code was implemented using resources from the beginning labs and assignment in WebGL focusing on model movement.

## Interacting Objects

* Walls
  + Summary of functionality
    - Collision detection for the player object, it prevents the player from moving in the direction of the wall.
  + Link to theory
    - Collision idea
  + Implementation details
    - Code snipet,
    - Inclusion of the stop field in the generation of each objects collision detection function.
* Finish line
  + Summary of functionality
    - Collision detection for the player object, it indicates that the player has reached the end of the game
  + Link to theory
    - Collision details
  + Implementation details
    - Code snippet, and the inclusion of another field called final for each collide able object

## Non-Player Character

* Summary of functionality
  + The enemy red tank, it moves in a fixed loop around its position.
* Link to theory
  + Creation of the model with its texture, and movement around the scene at fixed locations.
* Implementation details
  + Texture image and colour values retrieved from the scene.js file. Its then linked into the fragment and vertice shader where the Blinn-Phong model equation will be used to calculate colour. Texture colour is calculated

## Change of View

* Summary of functionality
  + Two types of views included; top-down and first-person.
* Link to theory
  + Perspective
  + Look-at vector
* Implementation details
  + Two cameras that are initialized within the main javascript file outside of the game.
  + Top down view and first person view are both initially set.
  + First person view follows the movement of the player within the scene.
  + Within drawscene function, camera positioning is indicated with a list index. This affords more flexibility within the game, and allows more camera views without having to include additional code.

## Additional Functionality

* Adding sound from projectiles
  + Summary of functionality
    - Sound file is loaded from the HTML with the audio tag and included with an id.
    - Within the game.js file inside the onPlay function, the specific id is retrieved and linked to a constant variable.
    - The sound is sped up to the maximum amount to allow for multiple firings.
    - Hitting the space key, which creates the projectile, will now also play the sound
  + Link to theory
  + Implementation details
* Adding basic timer
  + Summary of functionality
    - A built-in javascript world-timer without modification
    - It links to an id within the HTML file, and is constantly updated within the onUpdate function in game.js.
  + Link to theory
  + Implementation details
* Adding keymapping to html
  + Summary of functionality
    - The indicated keys are linked to their respective HTML tag within both the keyup and keydown listener.
  + Link to theory
  + Implementation details
* Music
  + Summary of functionality
    - Audio is added inside the HTML file using the audio tag.
    - Soundtrack’s are stored within a list, and the current index is stored within the game variable. The name of the song is linked inside a separate list and linked to an HTML header tag to display output to the user.
  + Link to theory
  + Implementation details
* Changing Lights on a counter
  + Summary of functionality
    - The global light is run through a progressive updater through each RGB colour.
  + Link to theory
  + Implementation details

# Analysis and discussion