3D TANK GAME ASSIGNMENT

EEE2041 COMPUTER VISION AND GRAPHICS - SPRING 2017

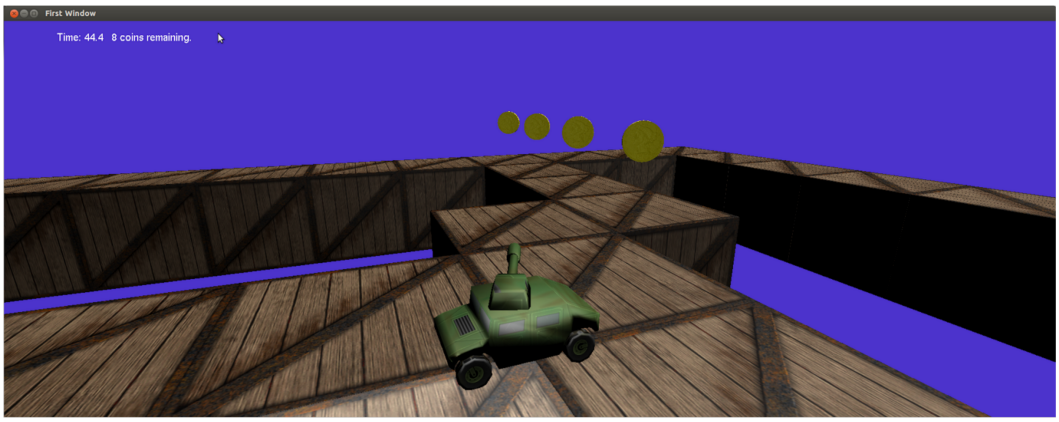

Figure 1: Example maze stage with user-controlled tank and target objects

1 INTRODUCTION

Your task is to develop a 3rd person maze action game using OpenGL. The goal of the game is to navigate a tank around a maze using the keyboard and mouse and eliminate all targets within a time limit while avoiding traps and falling off the edges. An outline of the functionality you are required to implement is given below. Marks will be awarded for providing additional functionality and for ease of user interaction.

Assessment will be made in two parts:

1. Demonstration of your working software in the Penguin Laboratory (32BB03) from 4-6pm Monday 15th May (week 11). This demonstration should last approximately 10 minutes per person, and show off what you have achieved.
2. A short report (5 pages – see section 3) detailing the methods you have implemented and results. To be submitted by 4pm on Tuesday 16th May via SurreyLearn.

Details of the requirements for implementation, the report and assessment are given below.

2 IMPLEMENTATION STAGES

Implement an interactive 3rd person maze action game using OpenGL with the following stages (further details for each stage are given in section 4):

1. Maze Environment: display the maze environment with targets and traps (see section 5.1). Implement game play by scoring the game based on the number of targets destroyed by a user controlled object moving around the maze.
2. Tank model: Load and display the tank model with texture maps (see section 5.2).
3. Tank Movement: control the tank to move around the maze through keyboard input. The tank turret should rotate with respect to the camera (see section 5.3).
4. Camera Control: control the camera to follow the tank (see section 5.4).
5. Additional functionality: Use shaders with lighting and material properties to enhance the appearance of the game. Give the scene greater realism and interactivity such as shadow rendering and textures. (see section 5.5)

3 REPORT

A short report of approximately five pages should be produced containing the following:

1. Title page - Title/author/module.
2. Introduction (~1/2 page) - summarise what the objective of the project was and techniques used.
3. Method (~3 pages) - Detail the techniques implemented for stages 1-5 of the implementation above + any additional methods used.
4. Results (~1 page) - Images showing rendering of the game.
5. Conclusions (~1/2 page) - What have you learnt from this exercise.

Reports should be submitted via SurreyLearn by the deadline given in section 1. Copies of your source code and makefiles (not executables) should be emailed to: a.hilton@surrey.ac.uk.

4 ASSESSMENT

The assessment will be based on both a demonstration of the project and the report. Marks will be awarded for completing each of the stages (1-4) of the implementation outlined above. Additional marks will be awarded for implementation of the more advanced features in steps 5 and innovative use of other techniques to enhance the rendered image or user interaction. The assignment together with lab. attendance will count towards 30% of the overall module assessment.

5 IMPLEMENTATION DETAILS

5.1 *Maze Environment*

1. The maze path is formed from a collection of floating tiles/blocks where the tank is allowed to move. A tile/block may be represented by a square or a cube on the x-z plane
2. The entire maze can be represented efficiently by an MxN integer matrix where 0 indicates empty space, 1 an empty block, and 2 a block with a target. Maze environments should be read from a file to allow for the creation of different levels:

0	0	0	0	0	0	0	1	1	1
2	2	2	2	0	0	0	1	0	0
1	0	0	1	0	0	0	2	0	0
1	0	0	1	0	0	0	1	0	1
1	2	1	1	0	0	0	1	0	1
0	0	0	1	1	1	1	2	0	1
0	0	0	1	0	0	0	0	0	1
0	0	0	1	1	2	2	2	2	2

3. Each block may hold a target object or a boost package which can be represented as a 3D shape (sphere, cube, cone etc.).
4. Tokens/targets can be collected either by shooting at them or moving the tank over them. This can be achieved by implementing basic collision detection. For example: given a point X, determine whether it lies within a sphere, cube etc.

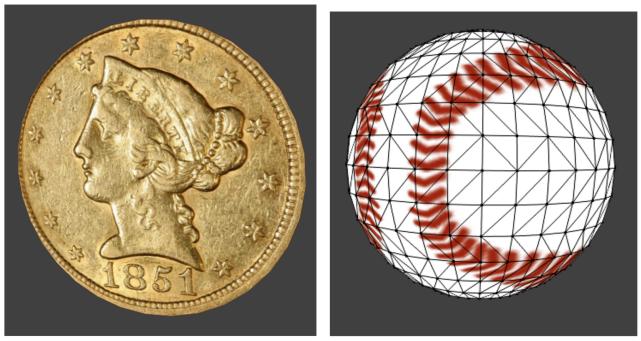

Figure 2: The provided textured coin and ball objects can be used as tokens to collect and object to be fired from tank. These can be found in the models directory.

5.2 *Tank data*

1. C/C++ Code for reading and drawing '*.obj' files in OpenGL can be found in previous lab exercises.
2. Tank Models: chassis.obj, turret.obj, front_wheel.obj, back_wheel.obj, tank.obj, can be found in the models folder
3. Texture image for models: tank.bmp.

Figure 3: Tank data: texture map hamvee.bmp (left); tank parts (middle); full-tank model (right)

5.3 *Tank Movement*

The tank can move in all directions in the x-z plane using keyboard keys while turning according to the direction of the movement. The wheels should rotate according to the distance travelled.

1. The position of a tank at any moment is determined by the speed and the direction of the movement given by:

$$P_{new} = P_{prev} + \Delta t \cdot V_{new} \tag{1}$$

$$V_{new} = n_{new} + u_{new} \tag{2}$$

   where n_{new} is a normalized vector (direction) and u_{new} is the magnitude of the velocity.

2. The tank should accelerate or decelerate using keyboard keys. The new velocity is given by:

$$V_{new} = V_{prev} + \Delta t \cdot (a_{accel} - a_{decel}) \tag{3}$$

3. If the tank's new position is outside the path boundaries, the tank should fall from the edges.

   Hint: You will need to implement a timer to update the tank's position. This can be done using the glutTimer function which has been set up for you in the starting code. It is also useful to implement key buffering to allow multiple keys to be registered

at once. A basic version of this has been provided and further details can be found in this tutorial.

5.4 *Tank Turret Movement*

1. The turret should rotate according to the rotation of the camera. This can be done using the SphericalCameraManipulator used in lab exercises. conisder using the glutPassiveMotion
2. The turret should fire projectiles that are affected by gravity using the same equations as above.

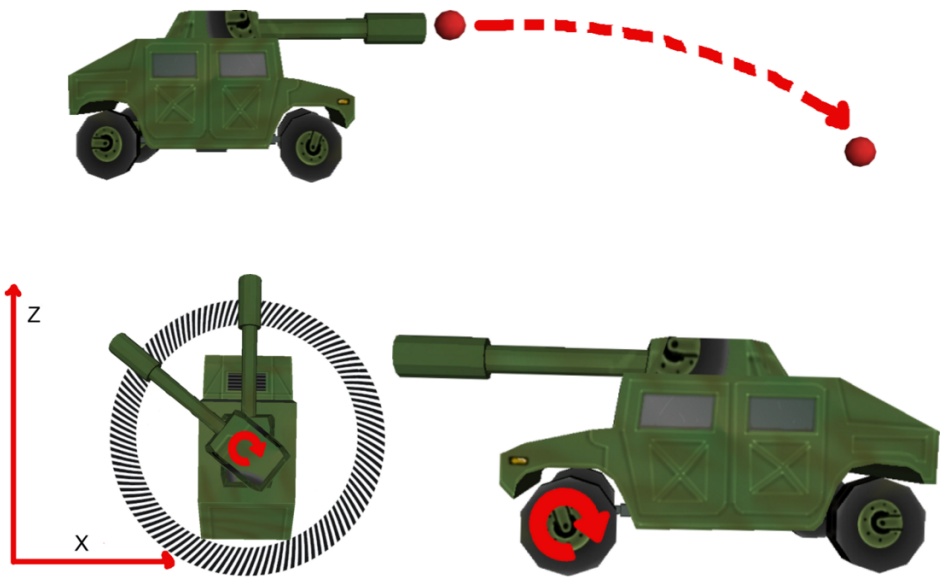

Figure 4

5.5 *Camera and HUD (Heads up Display)*

1. The camera should follow the tank as it moves around the maze from a fixed distance.
2. The time remaining and the number of tokens collected/destroyed should be shown on the screen. A simple function to draw 2D text is given in previous lab exercises.

5.6 *Other things that could be considered*

Additional Features to consider:

- randomise maze or allow maze to be loaded from file
- using alternative shaders
- make use of lighting effects

Other elements that can enhance the game experience. For example:

- Adding menus.
- Adding different levels of difficulty.
- Adding enemy tanks with artificial intelligence

EE2041 Assignment Assessment Sheet – Spring 2017

Student Name: URN:

Assessor Name: AH MV SF

Assignment: Tank Maze Game

1. Game Environment - course layout - game play		[15%]
2. Tank Model -load and display		[10%]
3. Tank Movement - movement control with key-board input - Turret movement		[15%]
4. Camera Control - control of the camera movement - Heads up Display		[15%]
5. Additional Features: - game-play: levels etc. - appearance: texture, scene, illumination, materials, shadows - motion control: physics - interaction control: track-ball - camera control: tank view		[15%]

Report [Note: failure to hand-in the report on-time will result in late submission penalties for the whole assignment]

Structure - Abstract/Introduction - Conclusions		[3%]
Methodology		[3%]
Results		[4%]

Lab. Attendance weeks 1 to 10		[20%]

Overall Grade (100%)	