COSC1187 – Interactive 3D Graphics & Animation

Semester 1 – 2021

Assignment 2 - ASTEROID ARENA 3D

Due Date: End of Week 13 - 11:59pm, Sunday 6th June, 2021

This assignment is worth 50% of your final mark for this subject. You must work on this assignment on your own and submit original code you have written yourself.

You will get a score between 0 and 100. The requirements each have different levels of difficulty. To get the highest marks, you must implement the most difficult version of each feature. You can implement features in any order. You don't have to choose the same difficulty for every feature.

You will be building a 3D version of Asteroid Arena, the game you developed for Assignment 1. You may re-use as much or as little of your original code as you wish.

Submissions will be via Canvas and MUST follow these guidelines:

- Please submit a single ZIP file containing your source code, README.PDF file and any resources needed by your game (such as texture maps or models).
- The file name must be a2_*<your student number>*.zip
- DO NOT SUBMIT OBJECT OR EXECUTABLE FILES!
- If you use a bitmap or 3d model that you find on the internet, you MUST check copyright and credit the original creator. List this info in the README.PDF file.
- If submitting on macOS or Linux, please include a shell script file named build.sh which will execute the appropriate command(s) to build your code for me.
- Late submissions will receive a 10% per day penalty.

MANDATORY

You must include a file named README.PDF which tells me which level of difficulty you have implemented for each feature, and a short description of the techniques you used for that feature. If any features are partially implemented, make sure to tell me here so that you get some marks for it!

The game is in 3D and uses Perspective projection.

Movement of all game entities MUST be calculated at every frame and MUST be based on time elapsed since the last frame. See "time delta" code from Tutorial exercises.

ARENA

The arena is now a 3D cube centred at the world coordinate system origin.

EASY (12) - Your ship can fly around inside the arena but will "die" if it collides with any boundary of the arena. Draw the arena walls as a light grey grid/mesh.

MEDIUM (16) - All of EASY, plus: If your ship gets within some distance from an arena wall, change the colour of the mesh to warn the player that they are about to collide with the wall and die. Only the one wall which your ship is approaching should change colour. The wall should change back to its original colour as soon as the player is a safe distance away.

HARD (18) - All of MEDIUM, plus: Implement a SkyBox with a texture map around the entire arena.

SPACESHIP MODEL

Your spaceship must be a 3D model made up of a set of triangles or quads.

EASY (6) - You must use at least two different surface materials which each define at least diffuse and specular material properties.

MEDIUM (10) - All of EASY, plus: Your ship should also be partially or fully texture mapped in addition to its material properties.

HARD (12) - All of MEDIUM, plus: To show your understanding of and ability to implement a hierarchical relationship between model sub-objects, your ship should have wings which change orientation depending on whether it is stationary or moving. It should transition/animate smoothly between these two states.

ASTEROID MODEL

EASY (6) - Your asteroids should be a perfect sphere which is smooth shaded and has material properties which show a specular highlight. You may use the GLUT functions for this.

MEDIUM (10) - All of EASY, plus: You may NOT use GLUT functions for drawing the sphere. Generate the asteroid vertices procedurally and specify vertex normals and add a texture map to your asteroid in addition to the material properties.

HARD (12) - All of MEDIUM, plus: Generate the asteroid vertices procedurally such that each vertex has some random amount of perturbation to make it look irregular (like in Assignment 1, but in 3D).

ASTEROID MOVEMENT

EASY (6) - Asteroids should spawn in a random place outside the arena and be launched towards the current player location at a random speed. Each asteroid should be a different size determined randomly when it is spawned. Spawn multiple asteroids in waves, like in Assignment 1. Asteroids move along their trajectory without any rotation. Asteroids do not collide with any other asteroids and they pass through the arena walls.

MEDIUM (10) - All of EASY, plus: Asteroids spin about their centre in a random direction and at a random speed as they move along their trajectory. Asteroids must bounce off arena walls.

HARD (14) - Asteroids spin about their centre in a random direction and at a random speed as they move along their trajectory. Asteroids must bounce off arena walls and other asteroids.

LIGHTING

EASY (2) - Enable OpenGL lighting and add a small amount of ambient lighting to the entire scene.

MEDIUM (4) - All of EASY, plus: Add a single directional light to the scene, and make sure that surfaces which have specular material properties show it.

HARD (6) - All of MEDIUM, plus: Add at least one animated point light source to the game. Animate its position and colour in some way.

BULLETS and SHOOTING

EASY (6) - Bullets should shoot out from the front of the ship and move in a straight line defined by the current ship movement direction. If a bullet hits an asteroid, the asteroid should disappear. Bullets can be drawn with any OpenGL primitive.

MEDIUM (10) - All of EASY, plus: Bullets should be made up of a single textured quad which is always facing the camera. The texture should have transparent regions. Asteroids should have a number of "hit points" calculated based on their size.

HARD (12) - All of MEDIUM, plus: The bullet texture should be animated, ie: made up of more than a single frame, with frames changed at a constant rate. The bullet quad should rotate about its centre as it is moving.

EXPLOSIONS

MEDIUM (4) - When an Asteroid or your Ship explodes, the original model should be replaced by a particle explosion like in Assignment 1, but instead of using POINTS your

particles should be triangles or quads. Randomise and animate the size, colour and other material properties, direction, rotation and speed of each particle.

HARD (8) - When an Asteroid or your Ship explodes, the original model should be replaced by a particle explosion like in Assignment 1, but instead of using POINTS your particles should be quads which always face the user and have a texture that has some transparent regions. The particles should be sorted based on z-order in camera space and blended with transparency while drawn back to front.

CAMERA and SHIP MOVEMENT

EASY (12) - Your ship movement model is completely up to you, but it must use a combination of Keyboard and Mouse controls. Your camera must be positioned somewhere behind the ship and may remain fixed at that position at all times. This ship may remain in a fixed location on-screen while you fly through space.

MEDIUM (16) - All of EASY, plus: Implement ONE of the following two features to give the player some way of seeing what's around them:

- Provide a mechanism for the player to look to the **Left**, **Right** and **Behind** the spaceship. This could simply consist of switching the camera to a side or rear view while an appropriate key is pressed down. The ship should not be rendered while the camera is displaying one of these camera views.
- Provide a mechanism to move the camera Forward and Backward in a straight line, while maintaining a position behind the ship and pointing in the same direction. It should never be possible to move the camera forward past the ship's location.

HARD (18) - Your ship movement model is completely up to you, but it must use a combination of Keyboard and Mouse controls. The camera should be positioned somewhere behind and above the ship (preferred position). When the ship changes direction in any direction, it should be animated onscreen to turn, roll or bank appropriately using smooth animation. Camera movement must be dynamic and animated while it "catches up" with the ship and positions itself in its preferred position. Think of the camera as being attached to the ship with an invisible string, and as the ship manoeuvres around, the camera smoothly follows along with a bit of lag - turning and moving until it reaches its preferred position back behind the ship.