Student ID: 27033004
Name: Jonathan Edwards

COSC363 Assignment 1 Report

The Scene

(narrator) The life of a museum guard is an easy life. (museum guard) “All I do is walk around some priceless science based objects all day, making sure nothing bad happens”. (narrator) He soon finds out that his job is in jeopardy when something unexpected happens by the cannon one fateful day.

Figure 1: Overview

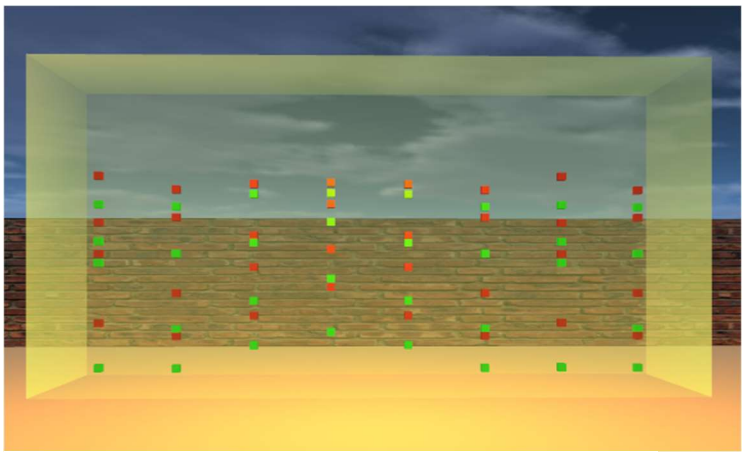


Figure 2: ‘Gravitational less’ cabinet which contains 8x8 cubes, each with infinite oscillation

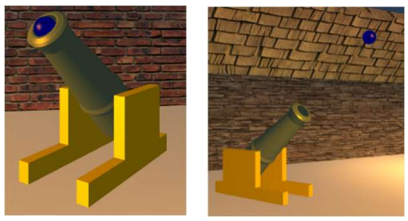


Figure 3: Cannon – stationary and non

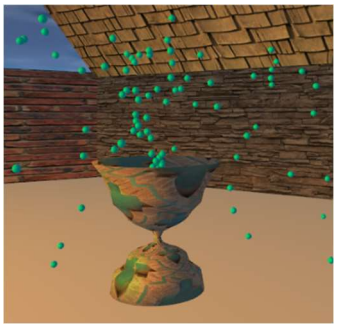


Figure 4: Soapy fountain

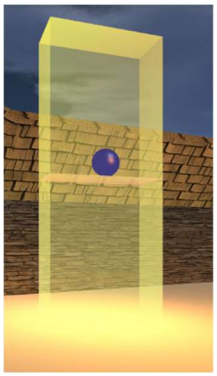


Figure 5: Magic bouncy ball

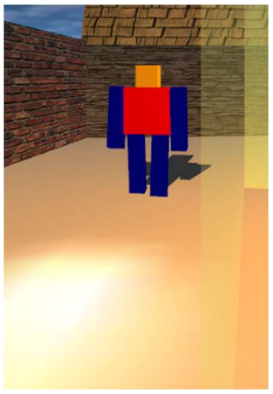
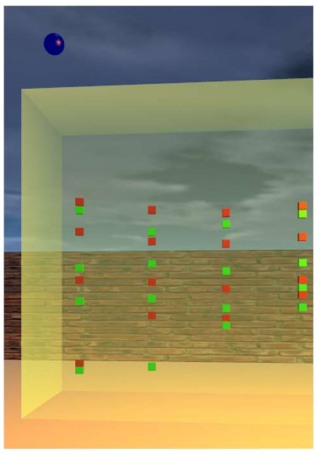
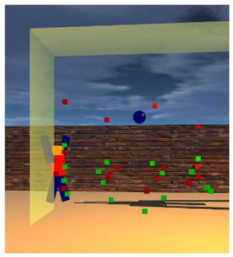
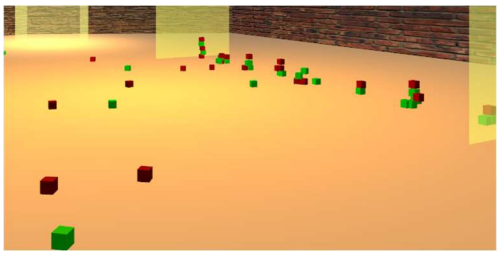
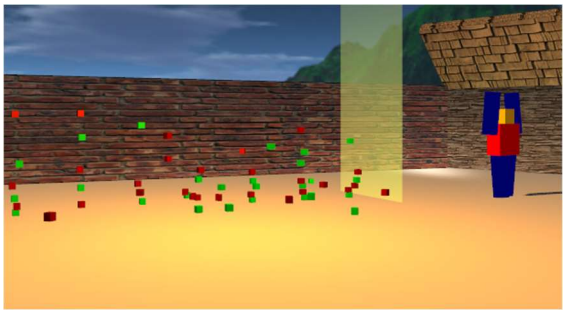


Figure 6: Guard – with flashlight and shadow









Figure(s) 7: Cannon ball breaking gravitational less cabinet – cubes are subjected to gravity

Student ID: 27033004
Name: Jonathan Edwards

Figure 8: Outside, rotating door and skybox

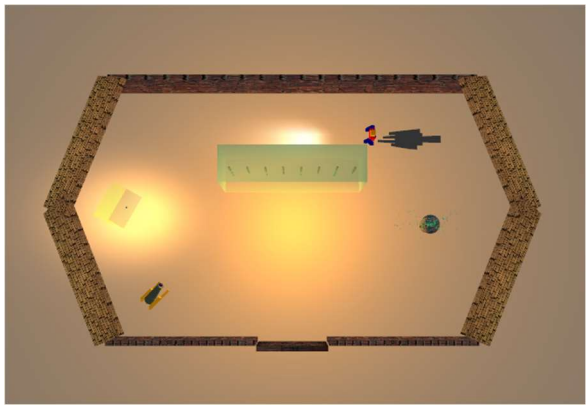


Figure 9: Top down view

Features – plus extra

1. **Skybox:** The whole cube scene is textured with textures given to us by the University of Canterbury. The actual museum scene is in the middle of the cube. This results in the edges of the cube to be invisible to the eye. (see fig 8)

2. **Door:** The door is textured with textures from https://www.textures.com/ all the ways round with a modern look – Extra feature.

3. **Building(Walls + Roof):** Designed using a simple 1^3 cube made by vertex coordinates around an axis of 1. Then by using simple trig created, I created a hexagon. The same Trig concept was also use with the roof. This is shown in (fig 9 above) – Extra feature.

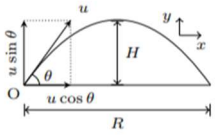
```
44   // move wall in place respect to angles
45   #define WALL_X      WALL_SCALE_LENGTH*sin(WALL_ROT_RAD)
46   #define WALL_Z      WALL_SCALE_LENGTH*cos(WALL_ROT_RAD)
```

```
1211    // front left
1212    glPushMatrix();
1213        glColor3f(1, 0.8, 0);
1214        glTranslatef(-WALL_X-WIDTH_SPACE, 0, 0);        // move left
1215        glRotatef(WALL_ROT_THETA, 0, 1, 0);
1216        wall();
1217    glPopMatrix();
```

Example of front left wall transformed using Trig movement

4. **Cannon:** Designed using glut objects.
.off file used from lab 2. When triggered, Cannon fires a sphere and moves in an arc shape by using projectial motion formulas (as sgown to the side) – sphere has collision detection with glass wall. (see fig 3)

Projectile Motion:

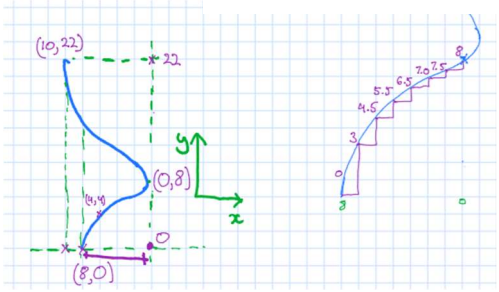


$$x = ut\cos\theta, \quad y = ut\sin\theta - \tfrac{1}{2}gt^2$$
$$y = x\tan\theta - \frac{g}{2u^2\cos^2\theta}x^2$$
$$T = \frac{2u\sin\theta}{g}, \quad R = \frac{u^2\sin 2\theta}{g}, \quad H = \frac{u^2\sin^2\theta}{2g}$$

https://en.wikipedia.org/wiki/Projectile

5. **Fountain (Custom built):** The fountain model was designed by using object modelling – sweep surfaces technique from lab 4 and 5. (see figs below). The texture used on this surface was from https://www.textures.com/ (see fig 4)

Fountain designed on tablet



Base plot – Model drawn in increments of 3.6°

```
1927   #define   N   20
1928
1929   float vx_init[N] = { 8, 7.5, 5.5, 5, 4, 3, 2, 1, 0.75, 0.5, 1, 2, 3, 4, 5, 6, 7.5, 8, 9.5, 11 };
1930   float vy_init[N] = { 0, 3, 4.5, 6.0, 6.7, 7.0, 7.5, 8.0, 8.4, 9.0, 10.0, 10.5, 11, 11.5, 12.2, 13.5, 15, 17, 20, 22 };
1931   float vz_init[N] = { 0 };
```

Student ID: 27033004
Name: Jonathan Edwards

6. **Particle System:** Water particles are shot off randomly in the 3d-plane. Gravity is applied to these to get a realism effect i.e.(curved shape). Projectile motion formulas used. There is a

```
#define MAX_PARTICLES   100
```

max number of particles that can be generated at every interval. The cycle of a particle is, Init at the start then loop through -> live -> die. At the end of die reset that same particle from the start. (see fig 4)

7. **Magic Ball - Challenge:** This ball can transform and reset to a different radius and mass size each loop.

```
double radiusBallBounce[5] = { 0.05, 0.1, 0.2, 0.25, 0.3};
double massBallBounce[5] = { 0.1, 0.5, 10., 30., 60.};
```

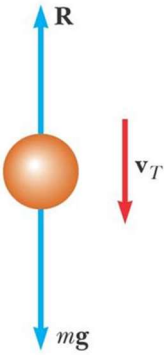
The V_Terminal(…) macro determines how much friction will be added to the system as the object collides. (see fig 5) (extra – physics based animation)

```
#define DRAG_CUBE   1.05  // face  , edge = 0.8
#define DRAG_SPHERE 0.45
#define AIR_DENSITY 1.225
```

Terminal Velocity

$$v_T = \sqrt{\frac{2mg}{D\rho A}}$$

m = mass
g = gravity(9.81)
D = drag coeff
p = air density
A = object area



Physics Uni Level 1 - Text book

8. **'Gravitational less' Cabinet - Challenge:**
Fig 2 is the centre of attraction with 0 gravity and collision detection between each cube in the 3 -axis. When Trigged by glass breaking as shown in fig 7. The Terminal velocity and projectile motion formula's are applied to each cube. This is implemented in the same way as the magic ball (above). The scene resets when a certain all objects are partially/fully stationary. – Multi/hyper threaded using pthread library for Linux. (disabled for Windows) (extra – physics based animation)
When the cannon ball hits the cabinet, the alpha value animates to value to 0. When alpha value = 0. (cabinet has been hit)

9. **Lighting and Shadow:**
   1. Room lighting = Lighting for the whole room at the centre. (warm orangey white color)
   2. Magic ball spotlight = A high contrast spotlight is displayed to the left. (as you walk into the room)
   3. Flashlight = Made from a spot light directed at an angle relative to the guard's position.
   4. Guard shadow = Shadow of the guard on the floor by using the Transformation Matrix.

10. **Two camera modes:**
    1. First person camera is the main camera to walk around the scene. (see fig 1)
    2. Top-down-view shows everything happening in the museum. (see fig 9) – Transforms whole scene around camera.

11. **FPS Counter – Extra feature:**
This counter keeps track of the whole scene speed as shown -> Using the glutTimerFunc(..) I set the FPS of the whole scene to "60" fps.

```
// FPS details
#define FPS             60
#define FPS_SEC         1000
#define TIMER_DELAY     FPS_SEC / FPS
#define QUATER_SEC      250
```

```
glutTimerFunc(TIMER_DELAY, idle, TIMER_DELAY);  // fps counter
```

```
// Display FPS every 1/4 a second
if (elapsedTime > QUATER_SEC) {
    sprintf(title, "Museum    FPS: %.2f", (frameCount / elapsedTime) * FPS_SEC);
    glutSetWindowTitle(title);

    endTime = startTime;
    frameCount = 0;
}
frameCount++;
```

Student ID: 27033004
Name: Jonathan Edwards

12. **Collision Detection:**
    1. Collision detection between the 3rd person camera (person) and the outside scene walls.
    2. Collision detection in Magic Ball (part 7)
    3. Collision detection in 8x8 cube array (part 8), Both collision detection between each cube in their column and the floor

```
682         else if ((ballPosX[z][z_2]-0.05) <= (ballPosX[j][z_2]+0.05) && (ballPosX[j][z_2]-0.05) <= (ballPosX[z][z_2]+0.05)) {
683             if ((ballPosY[z][z_2]-0.05) <= (ballPosY[j][z_2]+0.05) && (ballPosY[j][z_2]-0.05) <= (ballPosY[z][z_2]+0.05)) {
684                 if ((ballPosZ[z][z_2]-0.05) <= (ballPosZ[j][z_2]+0.05) && (ballPosZ[j][z_2]-0.05) <= (ballPosZ[z][z_2]+0.05)) {
```

8x8 (part 8) 3D - box collision detection

Control Functions

1. **V/v:** Toggle between the two cameras .
    - When top down view => use the "**+**" and "**–**" keys to adjust zoom.
2. **D/d:** Toggle front door to open/close.
3. **"Space bar"**: Fires the Cannon ball from the cannon.
4. ↑: Move the general camera forward.
5. ↓: Move the general camera backward.
6. ←: Turn the general camera left by 5 degrees.
7. →: Turn the general camera right by 5 degrees.

Program Development:

- Platform tested (OS) = Windows and Linux
- IDE = Visual studio code
- Multi-threaded for Linux, this feature is disabled in Windows (only use 1 core)
- Linux part 8 the "Gravitational less" cabinet has 2 worker threads to create the 8x8 for cubes and 2 worker threads for detecting collisions between each of them.

```
10  #define THREADS_BOX_COLL      2
11  #define THREADS_BOX_BOX_COLL  2
```

Build/Compile and run: (Inside the Museum folder)

- **Linux** = cd Linux && make && make
  "run0 – disabled threaded if computer has less than 2 cores" or
  "run1 – enables multithreading"
- **Windows** = cd Windows && make && make run
- **Note**: Do Not enable multithreading for CPU's with less than 2 cores (4 threads). Also connect power to Windows computer if FPS reads 30