**Programming Assessment**

**A REPORT**

***Submitted by***

**Beni Iyaka**

***In fulfillment of requirement***

***Of***

**Bachelor of Science (BSc.)**

**IN**

**COMPUTER SCIENCE**

**MACS**

**HERIOT WATT UNIVERSITY**

**F20GP**

**February 2016**

Table of Contents

Introduction\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1

Particle Explosion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1

Bouncing Ball\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6

Flocking Boids\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7

A \* Search\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8

**Introduction**

This coursework is based on developing different animated modules using OpenGL and C++. The models had to consist of the following:

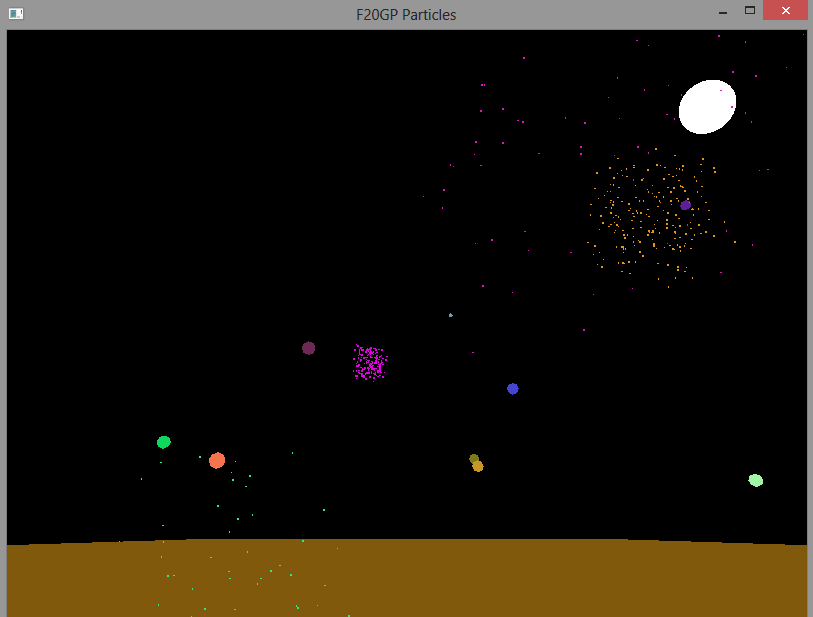
* Particle explosion
* Bouncing ball
* Flocking Boids
* Pathfinding

**Executive Summary**

1. **Particle Explosion**

For this model, I decided to make a firework system.

Figure 1 **Firework System**



For this model, I had to 5 different classes of which

One vector class: main.h

Four different classes: main.cpp, particles.cpp, random.cpp and sphere.cpp

**Vector Class**

**Main.h**

#include <cstdio>

#include <ctime>

#include <cmath>

#include <list>

#include <vector>

#include <random>

#include <iostream>

using namespace std;

#include <glut.h>

class Triple {

public:

float x, y, z;

};

class Sphere;

class Particle {

public:

Triple previousLocation;

Triple location, speed, acceleration;

Triple color;

int ttl;

void move();

Particle() {};

Particle(Sphere \*);

};

class Sphere {

public:

Particle center;

float radius;

int number\_of\_particles;

Sphere();

void move();

list<Particle \*> explode();

};

//Utils

double myRandom();

void add(Triple\*, Triple \*);

void set(Triple\*, float, float, float);

extern float SPHERES\_PER\_FRAME;

extern float GRAVITY\_POWER;

extern int SPHERE\_TTL;

extern int SPHERE\_PARTICLES;

extern int PARTICLE\_TTL;

This class handles the creation of sphere and randomly shooting them up on the screen and those spheres will then turn into particles and fall.

**Sphere.cpp**

This class handles the creation of small spheres and randomly setting the color and also moving it on the screen. Once they reach at the certain height, they explode and turn into small particles.

**Particle.cpp**

This class handle the creation of particles on the sphere explodes.

**Random.cpp**

This class handles the creation of random numbers that are then passed to sphere.cpp and particle.cpp.

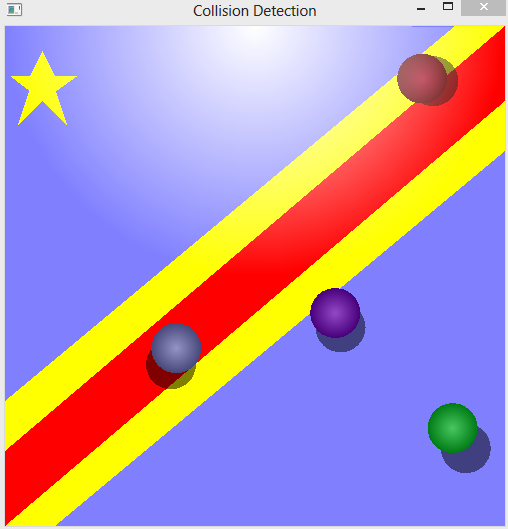
**Main.cpp**

This class handles the creation of the window size and also displays all the created classes on the window.

1. **Bouncing Ball**

For this module, I made four different balls moving around the screen and colliding with one another.

Figure 2 **Bouncing Balls**



For this model, I created two classes

One vector class: Vector.h

One other class: main.cpp

**Vector.h**

This handles the physics side of the program such as the collision process and also the layout of the circle.

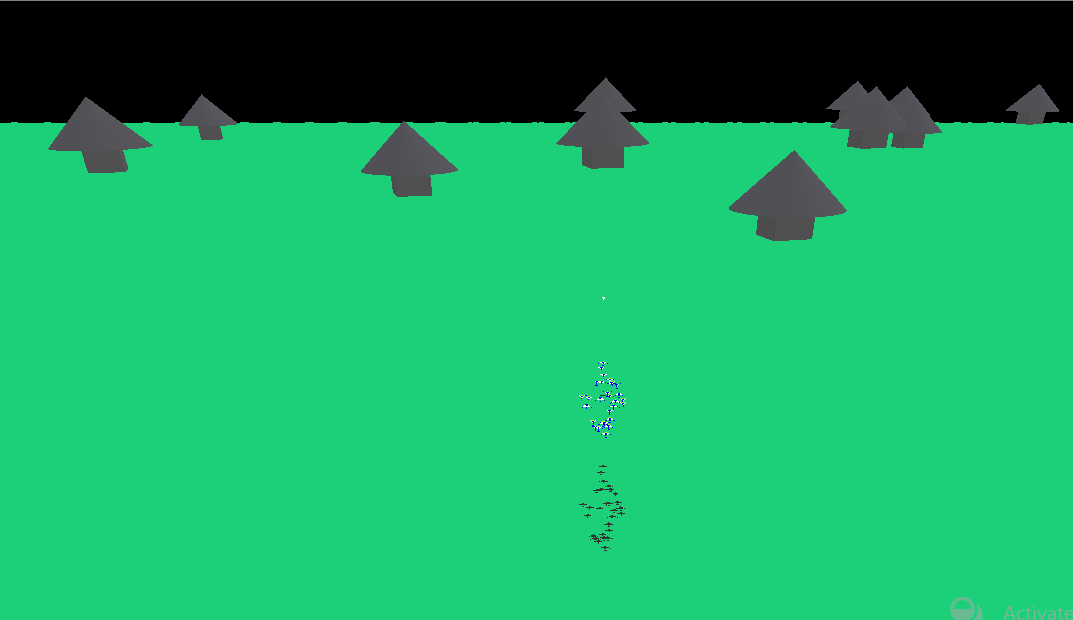
**Main.cpp**

This handles the background, the reshaping and also different functionalities of the program

1. **Flocking Boids**

For this model, I made flocks flying around and avoiding different buildings around them.

**Figure 3 Flocking Boids**

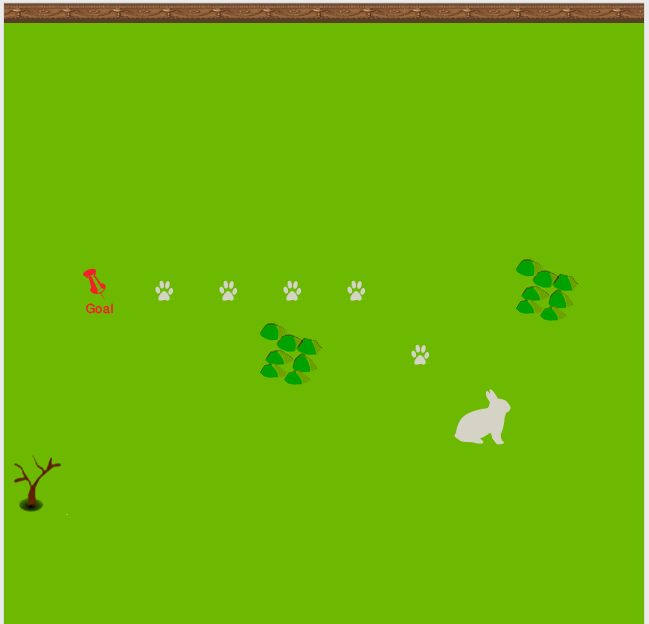
For this module, I made 6 classes of which each of them contain a header file to handle the physics part of the program such as collision avoidance and speed.

**Conclusion**

For the three models, I separated the physical behavior from visualization by creating a header (.h) file to handle all the physical behavior of the module and the .cpp files are used to handle the visualization part of each module.

**A\* Search**

Figure 4 Pathfinding



In this model, I made a rabbit trying to find food by searching for the appropriate path while avoiding obstacles.

This model is different from other types of search because it uses the heuristic method of which the rabbit discovers a certain path to reach the goal while avoiding different obstacles that it may encounter.