# Technical Document

**Project:** Ocean Ring Toss

**Author:** GameLoft Internship Team A

## Giới thiệu

Để vận dụng những kiến thức C++, 3D đã được học trong kì Gameloft Internship, Team sẽ xây dựng tựa game 3D Ocean Ring Toss trên nền tảng OpenGL ES 2.0 và Training Framework được Công ty Gameloft cung cấp.

Game Ocean Ring Toss mô phỏng sơ bộ lại trò chơi bắn vòng vào cột của tuổi thơ, với luật chơi đơn giản là đưa tất cả số vòng hiện có vào trong cột (mỏ neo).

### System requirements

* OS: Android 4.x and above
* Minimum Hardware:
* 1050 MHz Processor
* 256MB RAM
* 93MB storage available.
* Recommended Hardware:
* GHz Processor
* 128MB RAM
* OpenGL support, 96MB onboard video ram, pixel shader 1.1 supports, vertex shader 1.1 supports.
* 100MB storage available.

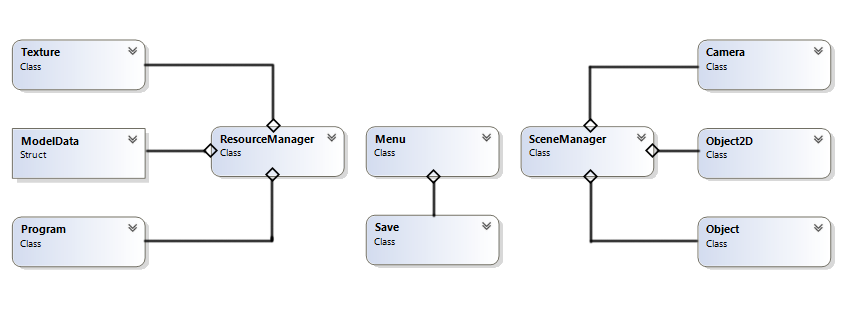
### Technical Issues

* Game could be slow on some phones.
* There are some random crashes in game.
* Graphics are blurry or crumble on some phones.
* Point checking sometime doesn’t work properly.
* Game might crash if users do not clear app properly.

### Third Party Tools

* Blender: a free 3D design tool.
* Pixlr: a free online tool for editing images.
* Visual Studio Community 2013: a free IDE tool for developing C++ program.

## Class Diagram



## Game Engine

Thực hiện công việc tập hợp tài nguyên và vẽ lên hình cho game. Game Engine bao gồm 2 thành phần chính là Resource Manager và Scene Manager.

### **Resource Manager**

**Phụ trách việc load tài nguyên** từ các file có sẵn theo những đường dẫn được khai báo trong file RM.txt, cũng như lưu trữ thông tin shader từ 2 file Fragment Shader, Vertex Shader. Tài nguyên gồm 2 phần chính là các file .obj và .tga.

**Các file .obj lưu trữ các thông tin** cơ bản của một **Model** bao gồm số lượng, tọa độ và thứ tự các đỉnh cần để tạo nên một Model.

**Các file .tga** lưu trữ thông tin texture.

### **Scene Manager**

Phụ trách việc dựng hình từ các tài nguyên đã được load sẵn qua Resource Manager**, tạo nên các cảnh chơi trong game.** Thông tin về vị trí, kích thước… của các mô hình được lưu trong file SM.txt .

Thông qua 2 thành phần này, việc dựng cảnh game chỉ cần thông qua việc sửa 2 file RM.txt và SM.txt .

## Gameplay

### **Vật lý**

Kỹ thuật chính của game là hệ thống Physic.

Team **sử dụng thư viện Bullet Physics để giải quyết yêu cầu vật lý** cũng như va chạm của game.

#### Thế giới Vật lý

Sử dụng btDynamicsWorld tạo nên thế giới vật lý và thiết lập trọng lượng cho nó thông qua hàm setGravity.

#### Áp dụng Cơ thể Vật lý (RigidBody) vào các Model trong game

Bullet Physics **cung cấp cơ thể vật lý (btRigidBody) với trọng lượng (mass)** cùng hình dạng vật lý (shape). Có nhiều dạng shape như Sphere, Cylinder, Box, TriangleMesh…

**Sử dụng hàm getOpenGLMatrix ta lấy được vị trí,** tọa độ của các rigidBody sau đó sử dụng các tọa độ này vẽ nên các Model tương ứng.

#### Xử lý Tính điểm

Trước tiên, **tạo một hàm CallBackFunc để xử lý va chạm**, xác định va chạm giữa vòng và cột.

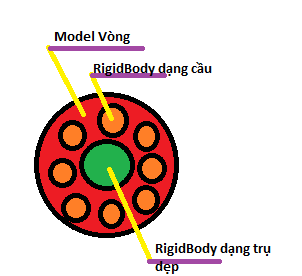
**Tạo một rigidBody** hình tròn đại diện cho **“lỗ” của vòng tròn**, **hai rigidBody hình trụ nằm trên trục của mỏ neo, một nằm giữa và một nằm trên đỉnh.** Khi “lỗ” va chạm với hai hình trụ này game sẽ tính điểm ra hoặc vào.

Vòng: bao gồm

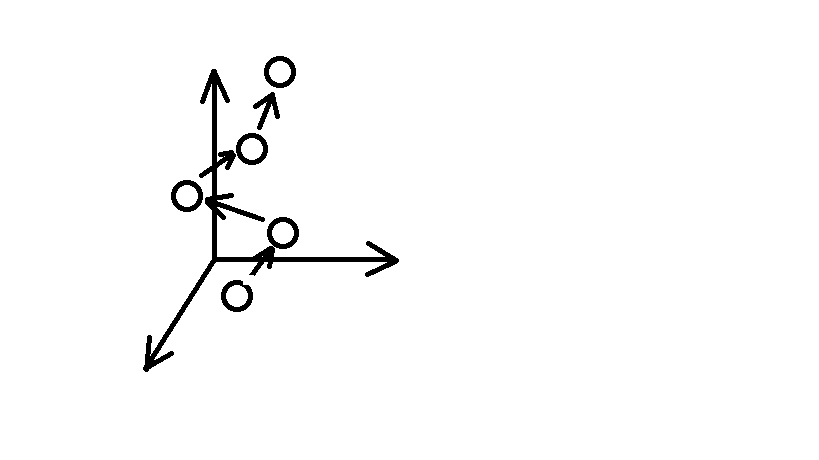
**-Model vòng.**

**-36 rigid body dạng cầu bao quanh** model vòng

-**1 rigid body dạng trụ dẹp lấp vào giữa lỗ hổng tren model vòng**, rigid body này vô hình và không có collision với vật thể khác ngoại trừ mỏ neo.



### Xử lý Bong bóng

* Tạo vector bubles biến toàn cục chứa object Bong Bóng.
* Hàm AddBubble : được gọi trong hàm AirGun. Mỗi lần gọi ,hàm thêm một số lượng object bubble vào vector bubbles.Mỗi bubble được tạo ra có vị trí bắt đầu bất kỳ trong một khu vực được quy định trước.
* Hàm UpdateBubble: được gọi ở mỗi vòng lặp game, thay đổi vị trí của mỗi Object bubble**. Các bubble sẽ di chuyển liên tục đi lên theo trục Y,** tuy nhiên bubble đồng thời cũng **chuyển động qua lại một cách ngẫu nhiên theo trục X,Z** góp phần tăng sự sinh động.
* 

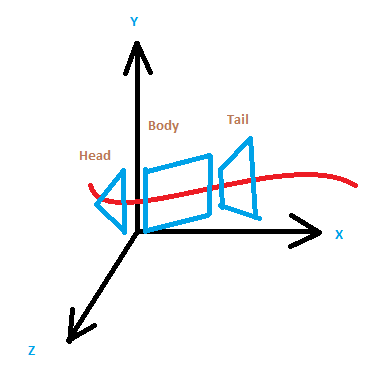
### Xử lý Chuyển động của Cá

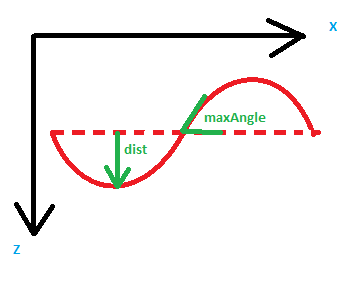
-Do tránh sử dụng Animation với Colada để giảm nhẹ lượng dữ liệu trong game nên đi đến quyết định sữ dụng code để xây dựng Animation.

-**Model cá được cắt làm ba phần: head, body, tail.** Ba phần được lưu trong một biến struck swimminFish. Biến swimminFish, tương tự bubble, được lưu trong một vector toàn cục.

-Hàm AddSwimminFishWithoutFuckingAnimation cho phép khởi tạo swimmingFish tùy theo model và texture mong muốn, đẩy biến swimmingFish vào vector.

-HàmUpdateSwimminFishWithoutFuckingAnimation: **dịch chuyển head, body, tail** của từng swimmingFish **theo một quỹ đạo hình sin trên mặt phẳng song song với mặt phẳng Oxz.**

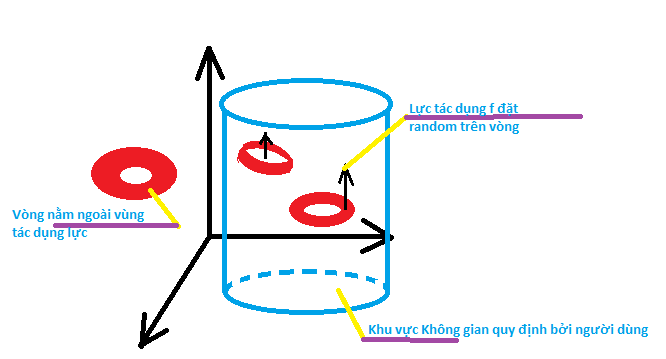




-Các thông số: dist, maxAngle, vận tốc di chuyển them trục Ox do người dùng tự chỉnh.

### AirGun

-Hàm AirGun: khi gọi, hàm sẽ **đặt một lực f lên vị trí random trên tất cả các vòng nằm trong khu không gian được quy định bởi người dùng.**

-**Vòng nằm càng cao-> lực tác dụng càng mạnh.**

### Sound

-Tụi em thiết kế 1 class trong Java,sử dụng 2 class là MediaPlayer và SoundPool là 2 class xử lý âm thanh trong Java.

### Accerometer

-Sử dụng kĩ thuật bắt sự kiện cảm biến trên thiết bị bằng những lớp được hỗ trợ Java.Trên các thiết bị thường có các loại cảm biến như : Type Accerometer,Rotation Matrix(tùy điện thoại hỗ trợ),Proximity,…Bằng cách sử dụng cảm biến Type Accerometer(hầu hết đều có trên tất cả các dòng điện thoại và phiên bản hệ điều hành Android) để làm GamePlay khi nghiêng điện thoại,những chiếc vòng sẽ di chuyển theo.

### Porting

-Vì game được viết bằng C++ chỉ chạy trên được window,mà Android sử dụng ngôn ngữ Java nên phải qua 1 bước là port sang Android.Khi port sang Android thì Jni sẽ là cầu nối để đóng gói những file code C++ thành 1 thư viện có đuôi là \*.so,sau đó trong Java sẽ gọi hàm để đọc thư viện này lên.

### Picking

Sử dụng kỹ thuật Ray Casting để xác định các nút bấm và kích hoạt súng bắn.