ICT215 Computer Graphics Principles and Programming, Semester 1, 2007
Project

Details:

Student Name : Stephen D'cruz
Student Number: xxxxxxx

Project Name: Remote Car on Pool Table
Proposition: To recreate the physics of a billiard table, and also to simulate the physics of car movement under different conditions.

Pre Production:

Concept Sketches:

Design:

Introduction:

First off, it was foolish of me to assume that I could simulate so many different aspects of the real world. Because of this I was only able to SIMULATE one real world situation, Simple one dimentional elastic collision between two objects of different masses. Other aspects have been EMULATED, using arbitary values to “emulate” the real world situations.

Emulated Aspects:

I would like to start with the “emulated” aspects first, as they do not simulate the real world, but they do have a small amount of physics calculations associated with them.

- Each object has friction:
  For each object a certain amount of friction is applied proportional to the mass of the object. This friction factor is also effected by the weather conditions.

- Weather Conditions:

2 Distinct weather conditions have also been modeled in game. Sunny, which is the default condition, and Raining (also included graphical Effect to show rain). When raining, the friction of objects to the ground decreases, and the handling of the car also decreases. Also the cars wheels spin faster, as if to loose traction on the ground.

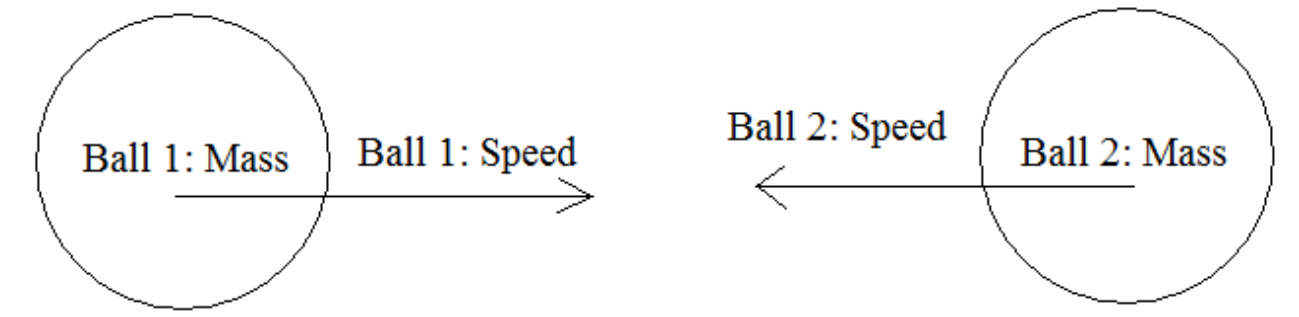
- Car Breaking:

The Final aspect of the "Emulated" situations is of the cars breaking. This will reduce the card speed based on it's mass, velocity, and the weather conditions.

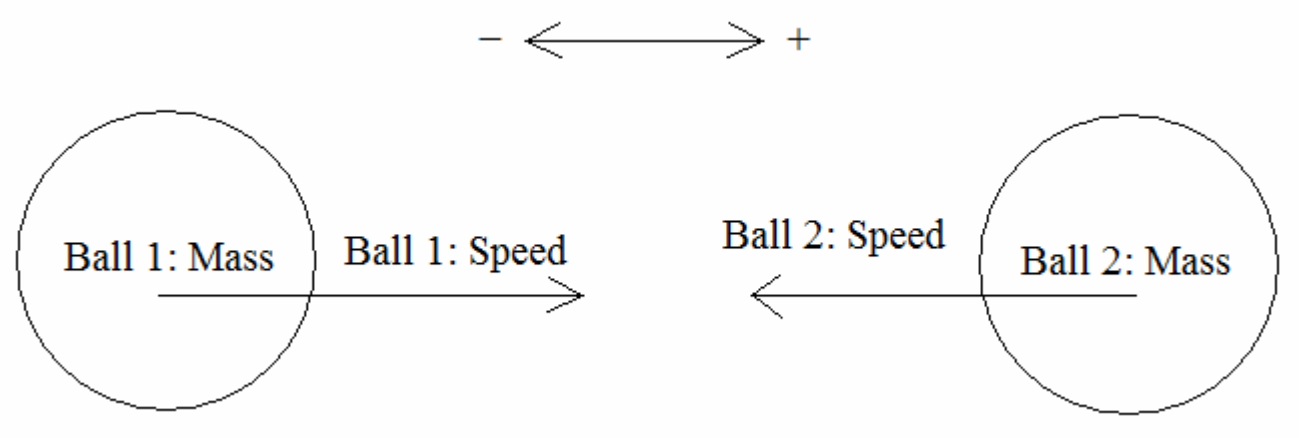
Simulated Aspects:

The Simulated aspect of the game is the One Dimensional Elastic collisions between objects of different masses. I did eventually calculate the equations for Two Dimensional Elastic Collisions, but found it too difficult to change the underlying code in the project.

Firstly to calculate the new velocity of the balls colliding, one needs to know some properties of the balls first:



Next we define which direction is Positive, in this case RIGHT is the Positive Direction.



After we have this we can then use the equation (P=MV), and Assuming Newtons Law of Conservation:

$$m_1u_1 + m_2u_2 = m_1v_1 + m_2v_2$$

where $m_1 = \text{mass of Object 1}$

$m_2 = \text{mass of Object 2}$

$u_1 = \text{Initial Velocity of Object 1}$

$u_2 = \text{Initial Velocity of Object 2}$

$v_1 = \text{Final Velocity of Object 1}$

$v_2 = \text{Final Velocity of Object 2}$

It is said that for a collision to be completely elastic all kinetic energy is conserved, thus using the equation ($KE = \frac{1}{2}mv^2$):

$$\frac{1}{2}m_1{u_1}^2 + \frac{1}{2}m_2{u_2}^2 = \frac{1}{2}m_1{v_1}^2 + \frac{1}{2}m_2{v_2}^2$$

where $m_1 = \text{mass of Object 1}$

$m_2 = \text{mass of Object 2}$

$u_1 = \text{Initial Velocity of Object 1}$

$u_2 = \text{Initial Velocity of Object 2}$

$v_1 = \text{Final Velocity of Object 1}$

$v_2 = \text{Final Velocity of Object 2}$

Using these 2 equations, one can solve simultaneously:

$$v_1 = \left(\frac{m_1 - m_2}{m_1 + m_2}\right)u_1 + \left(\frac{2m_2}{m_1 + m_2}\right)u_2$$

$$v_2 = \left(\frac{2m_1}{m_1 + m_2}\right)u_1 + \left(\frac{m_2 - m_1}{m_1 + m_2}\right)u_2$$

I then used these 2 equations in calculating the new velocity of the balls after collision:

```
earth.objects[i].angx = myobj.angx;
earth.objects[i].dx = earth.objects[i].x - myobj.x;//myobj.dx;
earth.objects[i].dz = earth.objects[i].z - myobj.z;//myobj.dz;

float v1 = myobj.vel;
float v2 = earth.objects[i].vel;

myobj.vel = (((myobj.mass - earth.objects[i].mass)/(myobj.mass +
earth.objects[i].mass))*v1) + (((2*earth.objects[i].mass)/(myobj.mass +
earth.objects[i].mass))*v2);

earth.objects[i].vel = (((2*myobj.mass)/(myobj.mass +
earth.objects[i].mass))*v1) + (((earth.objects[i].mass - myobj.mass)/(myobj.mass
+ earth.objects[i].mass))*v2);
```

NOTE: To emulate the presence of two dimensional collision the new directional vector of the object is calculated to be the distance between the two objects origins. This will give a fairly accurate result as the new direction.

USER GUIDE

Installation and Running

1. Open the CDROM Directory
2. Open Dev-Cpp, and run devcpp.exe
3. Then go to File->Open and select “CDROM:\Application\Project.dev”
4. Compile and Run

Operating the Simulation

Key	**Effect**
UP ARROW	Increase Car Acceleration
DOWN ARROW	Decrease Car Acceleration
LEFT ARROW	Turn Car Left
RIGHT ARROW	Turn Car Right
SPACE KEY	Handbrake for car. (Emulated)
‘c’ KEY	Turn ON/OFF Collision Detection
‘r’ KEY	Change the weather conditions (Emulates handling and friction of ground)
‘v’ KEY	Change the View of the Camera (View 1: Free Look) (View 2: 3rd Person – above car) (View 3: GTA View)
THESE KEYS ONLY APPLY IN CAMERA VIEW 1 (Free Look)	
‘w’ KEY	Move the Camera Forward
‘s’ KEY	Move the Camera Backward
‘a’ KEY	Move the Camera Left
‘d’ KEY	Move the Camera Right
SHIFT KEY	Double Camera Moving Speed