“BIVI” – GAME FOR SIMULATING THE MECHANICAL COLLISIONS

Victor ȚURCANU,
scientific advisor Mihail KULEV
Technical University of Moldova

Abstract: *This is a game which simulates the plain mechanical collisions of the bodies according to their mass and shape. There are approached three kinds of collision: circle to circle, circle to rectangle and rectangle to rectangle collision. It has two difficulty levels. At the beginning of each level there are explained the rules and the changes in gameplay. The player uses its mouse in order to move the target. The application was performed using the Processing programming language. According to the problem solution, the program was divided into eight parts.*

Keywords: *Computer game, mechanics of rigid bodies, plain collisions, Processing programming language.*

1. Introduction

One of the main applications of the computer programming is for simulation of the real life processes and their analysis. It is cheaper to make a program that will simulate an experiment than to perform it and spend a lot of money on expensive equipment or solutions. It is not necessary to buy and use the equipment, the only thing needed is their properties and behavior.

In mechanics, one of the most fascinating processes is the collisions of the bodies, especially spheres' collisions. It is easy to determine the direction and the velocity of motion of two bodies after collision, but what if, for example, there are twelve of them?

Problem: It needs to simulate body collisions when their number is greater than two.

Task: To create an application that will simulate collisions of balls which have different mass and different sizes, to compute and set their final velocity vectors and to visualize the process.

In the application will be treated all these situations: plane surface shaped objects collision, collision between a plane surfaced body and a spherical one, and the collision between two spherical objects. Moreover, it will be taken into account also the mass of the objects.

As a conclusion, the application should simulate:

1. The plane collision of two plane surface shaped objects.

2. The plane collision of a plane surface shaped object with a spherical object.

3. The plane collision of two spherical objects.

For making a Windows application, it will be needed an IDE – Integrated Development Environment. An IDE is a software application which provides comprehensive facilities to computer programmers for software development. It will be used Processing programming language [1] for simulation and visualizing the mechanical processes and because of this it will be used the Processing Software.

The Processing software runs on the Mac, Windows, and GNU/Linux platforms. With the click of a button, it exports applets for the Web or standalone applications for Mac, Windows, and GNU/Linux. Graphics from Processing programs may also be exported as PDF, DXF, or TIFF files and many other file formats. Future Processing releases will focus on faster 3D graphics, better video playback and capture, and enhancing the development environment. Some experimental versions of Processing have been adapted to other languages such as JavaScript, ActionScript, Ruby, Python, and Scala; other adaptations bring Processing to platforms like the OpenMoko, iPhone, and OLPC XO-1.

In Processing all projects are called sketches because, as its authors mention: “It is necessary to sketch in a medium related to the final medium so the sketch can approximate the outcome. Painters may construct elaborate drawings and sketches before executing the final work. Architects traditionally work first in cardboard and wood to better understand their forms in space. Musicians often work with a piano before scoring a more complex composition. To sketch electronic media, it's important to work with electronic materials. Just as each programming language is a distinct material, some are better for sketching than others, and artists working in software need environments for working through their ideas before writing final code. Processing is built to act as a software sketchbook, making it easy to explore and refine many different ideas within a short period of time”[2].

2. Theory used for solving the given problem

1) Plane collision of two spheres

According to the Second Law of Thermodynamics, the total energy before the impact and after the impact in the case of the elastic collision is the same, moreover – the sum of the kinetic energies of the bodies is the same as the energy spent on heating the bodies can be neglected. In these conditions, according to the same law of thermodynamics, the sum of the impulses of the bodies before and after the collision remains unchanged.

In order to determine the final vectors, we should compute their absolute value. If we denote with v_1 and v_2 the initial velocities of the first and the second body, and with m_1 and m_2 their respective masses, then:

$$u_1 = \frac{v_1(m_1 - m_2) - 2v_2m_2}{m_1 + m_2} \tag{1}$$

$$u_2 = \frac{2v_1m_2 - v_2(m_2 - m_1)}{m_1 + m_2} \tag{2}$$

where, u_1 and u_2 are the final velocities of the bodies [3].

2) Rectangle to rectangle collision

As it can be seen in the Fig. 1, two rectangles collide when they share a common piece of area. It can be detected by taking the components of the distance vector of the centers of the rectangles and compare whether they are smaller than the sum of the half of the widths and half of the heights. If so, according to four cases (from above, bottom, left and right) it can be set the new positions of the rectangles as it is shown in the Fig. 1:

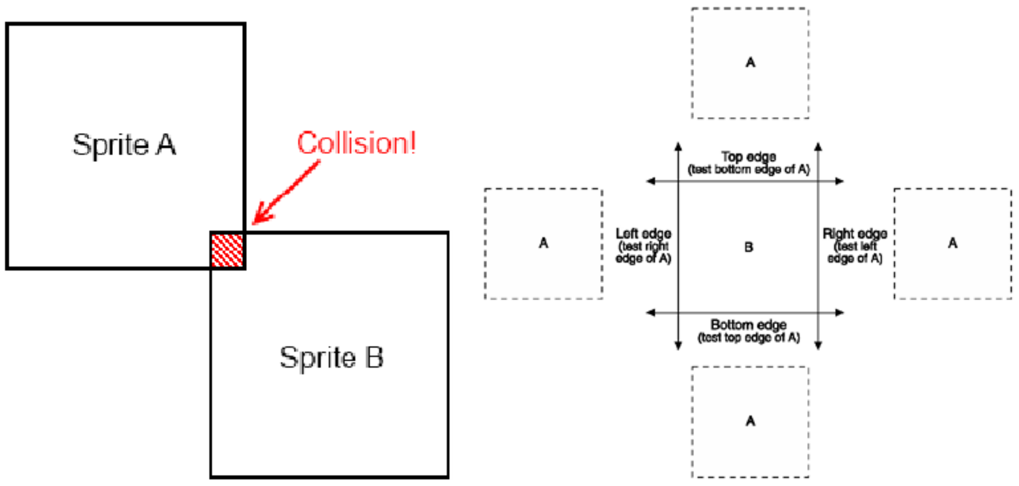

Sprite A
Collision!
Sprite B
A
Top edge (test bottom edge of A)
A
Left edge (test right edge of A)
B
Right edge (test left edge of A)
A
Bottom edge (test top edge of A)
A

Fig. 1

3) Circle to rectangle collision

And finally, the most interesting part and anti-intuitive is the circle to rectangle collision. Depending in what quadrant is the circle relating to the position of the rectangle, it can change its direction respectively. As it can be seen in the Fig. 2, the circle may be in one of the eight quadrants. If it is in one of the even quadrants, it is easy for program to change the velocity direction just by multiplying the specific component of the velocity vector by -1. As for odd quadrants, the velocity vector changes symmetrically according to the normal vector of the oblique line of the given corner.

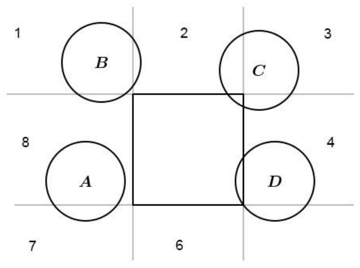
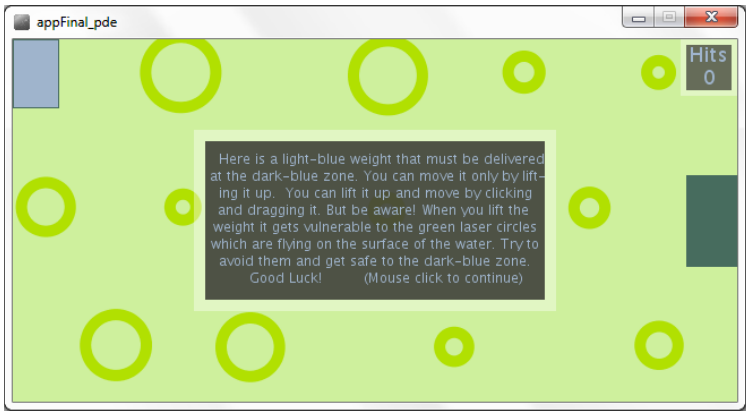


Fig. 2

3. Software description

The problem of simulation of mechanical processes was transformed in a game. The main reason why it was done is because while playing the user can better analyze the circumstances in which the collision occurs: in the same time he intuitively appreciates the correctness of the implementation algorithm and in the same manner he joins in the atmosphere of the game.

In this manner, it was created a legend of a weight that is at the bottom of a pool and which should be delivered to the safe zone (Fig. 3, first image). In the same time, at the surface of the water some laser circles which has different masses (which depends on their charges) and different radii, collide between themselves according to the mechanical laws of conservation of the impulse and of conservation of the mechanical energy.



Here is a light-blue weight that must be delivered at the dark-blue zone. You can move it only by lifting it up. You can lift it up and move by clicking and dragging it. But be aware! When you lift the weight it gets vulnerable to the green laser circles which are flying on the surface of the water. Try to avoid them and get safe to the dark-blue zone. Good Luck! (Mouse click to continue)

Fig. 3

In order to move the weight the player should lift it up. When it's lifted up, it gets vulnerable to the laser circles and therefore the player should avoid them. If the player lifts the weight while above it is a laser circle, then after some milliseconds it dies (Fig. 3, second image). The same happens if he got hit five times by the laser circles while he moves the weight. In the right top corner is displayed the number of hits (Fig. 3).

When it's finished the task, the player is proposed to join the next level (Fig. 3, third image). Here are some changes: firstly there appeared some walls through which the weight can't pass. Moreover, these walls are connected to the electricity source, so that if the weight touches them, it got electrified and can't hold it anymore. The player should lift it one more time in order to move.

4. The structure of the program

The program consists of eight blocks, among which it was divided the functions of the program. Each block represents a file with the extension *.pde which is a processing file.

The list of files and their contents:

appFinal – contains global variables and the main functions: **setup()** and **draw()**;

MouseOp – contains the function which operate with mouse: **mousePressed()**, **mouseDragged()** and **mouseReleased()**;

Safety – contains the function: **checkIfSafe()** and **checkWallCollision()**;

boundaryCollision – contains the function: **checkBoundaryCollision()**;

circleCollisionF – contains the function: **checkObjectCollision();**

classes – contains the declaration of the Rectangle and Ball classes;

messages – contains all functions that display the messages on the screen: displayStartMessage(), displayStartMessage2(), displayWinMessage(), displayLoseMessage(), displayHits() and displayNextLevel();

newRoll – contains the function checkRectanColl().

This program is used for easing the simulation of the mechanical processes, but in the same manner following the natural laws of dynamics of the rigid bodies.

5. Bibliography

1. *Official website of Processing* –*http://processing.org/learning/basics/setupdraw.html* (accessed 15.10.2013).
2. PETERS, Keith. *Processing. Creative coding and computational art* , Apress, 2008, 841 p.
3. PEIELRS, R. E.. *The Laws of Nature*, Charles Scribners Sone, 1956, 503 p.
4. SHIFFMAN, Daniel. *Learning Processing*, Elsevier, 2008, 472 p.