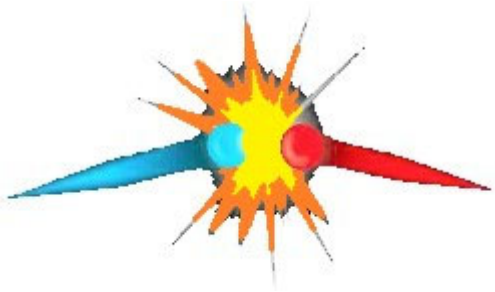


24-780 B—Engineering Computation

Assigned: Wed. Sept. 7, 2022
Due: Tues. Sept. 13, 2022, 11:59pm

Problem Set 2: Dice Rolling & Collisions

PS2-1 Dice Rolling Monte Carlo (40 pts)

A Monte Carlo Simulation involves the repeated use of random sampling to obtain quantifiable results of stochastic events. In particular, Monte Carlo methods allow for the inclusion of probability distributions for coupled and uncoupled events, thus making it possible to quantify systems whose exact behavior is not clearly understood. Often, Monte Carlo methods reveal that systems that appear random can have predictable behavior in the long run.

For this assignment, we are modeling a simple two-event problem to numerically quantify the results of rolling two dice. Each die has 6 faces, numbered 1 to 6 and each face is just as likely as any other to land on top when rolling. The result of each roll is the sum of the numbers on top of both dice (i.e., minimum 2, maximum 12). The two dice are independent events since the probability of one die's results is not affected by the other.

You are asked to write a program that asks the user to input the number of rolls they want to model and then perform the simulation while keeping a count of the results (e.g., how many times was a nine rolled?). Then, the program outputs the aggregated results of the simulation, which must include some sort of visualization. Since we have not done graphics yet, the visualization is limited to text.

Hint: What data structure can be used to store the count of all rolls as the simulation progresses?

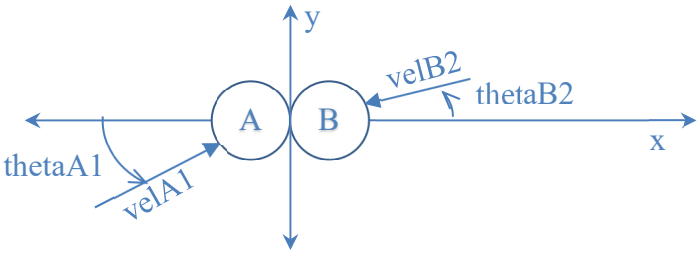
Sample output (feel free to "make it your own"):

```
============ DICE ROLLING ==============
Enter number of rolls > 50

Results (each '*' represents 1%):
   2: ******
   3: **
   4: ************
   5: ****************
   6: ********
   7: *************************
   8: **************
   9: **
  10: ******
  11: ******
  12: **
```

PS2-2 Collision Problem Quiz (60 pts)

Write a program that randomly creates a *unique* collision problem of the type shown in the attached example. Allow your program to generate reasonable random values for the problem parameters (masses, velocities, direction angles, coefficient of restitution, etc.). Then, present your problem to the user and ask for the resulting final velocities. Assume that the user has access to the following diagram showing all parameter labels and directions:



Where:

- Disc A has mass massA and is traveling at velocity velA1, at an angle thetaA1 from the x-axis (which is perpendicular to plane of impact and passes through the centers of the discs).
- Disc B has a mass massB and is traveling at velocity velB1, at an angle thetaB1 from the x-axis
- The collision has a coefficient of restitution of value eRes.

Let the value of the masses be a uniformly distributed random number varying from 0.5 to 4 kg, rounded to one decimal place. Similarly, let the angles vary from -60 to 60 degrees, rounded to zero decimal places, the velocities vary from 0 to 5 m/s, rounded to one decimal place, and the coefficient of restitution vary from 0.6 to 0.9, rounded to two decimal places.

Your program should ask the user to enter each of the 4 answers (i.e., velAx2, velAy2, velBx2, and velBy2) and determine if the answers are correct (perhaps also providing a measure of error as compared to correct answer). Use the attached sample problem to get a feel for how to use dynamics to work through the solution. You can also "hard wire" the inputs from the sample problem to test the accuracy of your program.

If you feel that your dynamics/physics background is limited enough to make the non-programming part of this problem too much to bear, let me know and we'll go over it step by step. It all comes down to working with systems of equations.

Deliverables

2 files, very appropriately named:

ps02_dice_yourAndrewID.cpp

ps02_collision_yourAndrewID.cpp

Upload the files to the class Canvas page before the deadline (Tuesday, Sept.13, 11:59pm).

Hint: Even if you name your file appropriately, be sure to include your full name within the code itself (perhaps as a comment block at the top of the file). It is also appropriate to include date and course info, plus a short description of what the program does. (Think about what part of the text in the assignment write-up can be copy/pasted or adapted for this purpose.)

Learning Objectives

Developing simple algorithms

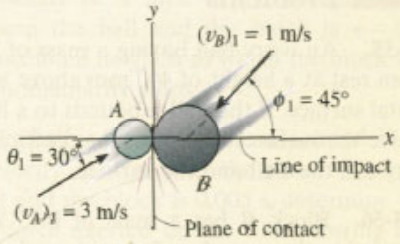
Making use of console input/output

Using functions effectively

Searching references (online and/or textbook) for C++ library functions.

EXAMPLE 15–11

Two smooth disks A and B, having a mass of 1 kg and 2 kg, respectively, collide with the velocities shown in Fig. 15–18*a*. If the coefficient of restitution for the disks is $e = 0.75$, determine the x and y components of the final velocity of each disk just after collision.



(a)

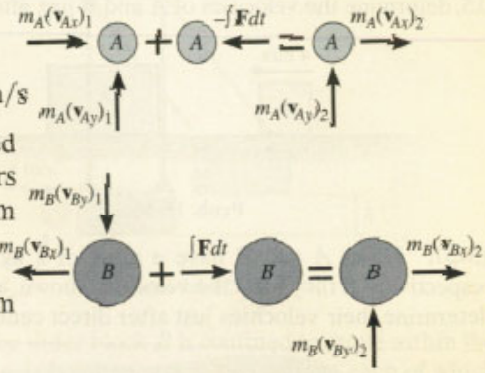
Solution

The problem involves *oblique impact*. Why? In order to seek a solution, we have established the x and y axes along the line of impact and the plane of contact, respectively, Fig. 15–18*a*.

Resolving each of the initial velocities into x and y components, we have

$$(v_{Ax})_1 = 3\cos 30° = 2.60 \text{ m/s} \qquad (v_{Ay})_1 = 3\sin 30° = 1.50 \text{ m/s}$$
$$(v_{Bx})_1 = -1\cos 45° = -0.707 \text{ m/s} \qquad (v_{By})_1 = -1\sin 45° = -0.707 \text{ m/s}$$

The four unknown velocity components after collision are assumed to act in the positive directions, Fig. 15–18*b*. Since the impact occurs only in the x direction (line of impact), the conservation of momentum for *both* disks can be applied in this direction. Why?



(b)

Conservation of "x" Momentum. In reference to the momentum diagrams, we have

$$(\pm\rightarrow) \qquad m_A(v_{Ax})_1 + m_B(v_{Bx})_1 = m_A(v_{Ax})_2 + m_B(v_{Bx})_2$$
$$1 \text{ kg}(2.60 \text{ m/s}) + 2 \text{ kg}(-0.707 \text{ m/s}) = 1 \text{ kg}(v_{Ax})_2 + 2 \text{ kg}(v_{Bx})_2$$
$$(v_{Ax})_2 + 2(v_{Bx})_2 = 1.18 \qquad (1)$$

Coefficient of Restitution (x). Both disks are *assumed* to have components of velocity in the $+x$ direction after collision, Fig. 15–18*b*,

$$(\pm\rightarrow)\ e = \frac{(v_{Bx})_2 - (v_{Ax})_2}{(v_{Ax})_1 - (v_{Bx})_1}; \qquad 0.75 = \frac{(v_{Bx})_2 - (v_{Ax})_2}{2.60 \text{ m/s} - (-0.707 \text{ m/s})}$$
$$(v_{Bx})_2 - (v_{Ax})_2 = 2.48 \qquad (2)$$

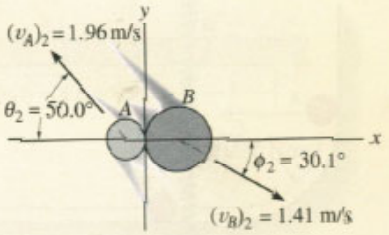
Solving Eqs. 1 and 2 for $(v_{Ax})_2$ and $(v_{Bx})_2$ yields

$$(v_{Ax})_2 = -1.26 \text{ m/s} = 1.26 \text{ m/s} \leftarrow \qquad (v_{Bx})_2 = 1.22 \text{ m/s} \rightarrow \quad \textit{Ans.}$$

Conservation of "y" Momentum. The momentum of *each disk* is *conserved* in the y direction (plane of contact), since the disks are smooth and therefore *no* external impulse acts in this direction. From Fig. 15–18*b*,

$$(+\uparrow)\ m_A(v_{Ay})_1 = m_A(v_{Ay})_2; \qquad (v_{Ay})_2 = 1.50 \text{ m/s} \uparrow \quad \textit{Ans.}$$
$$(+\uparrow)\ m_B(v_{By})_1 = m_B(v_{By})_2; \quad (v_{By})_2 = -0.707 \text{ m/s} = 0.707 \text{ m/s} \downarrow \quad \textit{Ans.}$$

Show that when the velocity components are summed, one obtains the results shown in Fig. 15–18*c*.



(c)

Fig. 15–18