Joon Ki Hong
MATH 2605
Project 1 – Pinball Project
Report

Implementation:
For this project, I used vPython 2.7. The main reason for this was mainly for the visual module included in vPython to visually demonstrate the reflections and paths. Code wise, I approached this in a linear fashion instead of using an OOP approach typically used in Java. The reason for this was mainly for simplicity and time constraints. Using the provided project document, I made the methods necessary (intersect, dotProduct, reflect) to carry out the simulation. I then created, using the visual module, 3 spheres at the locations provided in the document with the provided radius. I then created a smaller sphere at the origin (vector(0,0,0)) with a small radius (for visual purposes, does not affect the calculations). The code required to carry out the trials linearly was the most complex and thought provoking part of the assignment. Essentially I used two nested while loops. The outer loop would loop for the number of trials or in other words, the number of times the particle is shot. The inner loop would loop until the particle escapes the 3 spheres. Within the inner loop I updated the path of the particle using the visual module (curve() method) and placed small white spheres at the reflection points to indicate reflections. I also used 3 arrays to keep track of how many times a certain particle reflected against the spheres, the sequence of reflections, and the initial angles. The outer loop would reset the needed variables to reset the conditions necessary for the next trial. To implement the random angle or the systematic angles, I had to select either and comment out the lines necessary for either to operate (these lines are commented within the outer while loop). I also included a method (rate() is within the inner while loop), which is a part of vPython, that controls the speed in which the calculations are done so to slow down the animation.

1. **Random Trials:**
   Case: s=6 r=1
   List of frequency = [719, 241, 23, 17,0,0,…0]
   Trials: 1000
   Relative Frequency of 0: 719/1000 = 0.719
   Relative Frequency of 1: 241/1000 = 0.241
   Relative Frequency of 2: 23/1000 = 0.023
   Relative Frequency of 3: 17/1000 = 0.017
   Longest Orbit = 3 hits

   Case: s=6 r=2
   List of frequency = [411, 186, 158, 120, 67, 26, 19, 7, 3, 3, 0, 0,…0]
   Trials: 1000

Relative Frequency of 0: 411/1000 = 0.411
Relative Frequency of 1: 186/1000 = 0.186
Relative Frequency of 2: 158/1000 = 0.158
Relative Frequency of 3: 120/1000 = 0.120
Relative Frequency of 4: 67/1000 = 0.067
Relative Frequency of 5: 26/1000 = 0.026
Relative Frequency of 6: 19/1000 = 0.019
Relative Frequency of 7: 7/1000 = 0.007
Relative Frequency of 8: 3/1000 = 0.003
Relative Frequency of 9: 3/1000 = 0.003

Systematic Trials:
Case: s=6 r=1
List of Frequency = [720, 250, 17, 12, 1, 0, 0,…0]
Trials: 1000
Relative Frequency of 0: 720/1000 = 0.72
Relative Frequency of 1: 250/1000 = 0.25
Relative Frequency of 2: 17/1000 = 0.17
Relative Frequency of 3: 12/1000 = 0.12
Relative Frequency of 4: 1/1000 = 0.001
Longest Orbit = 4 hits

Case: s=6 r=2
List of frequency = [413, 192, 157, 116, 62, 31, 14, 8, 3, 3, 0, 0, 0, 0, 1, 0, 0,…0]
Trials: 1000

Relative Frequency of 0: 413/1000 = 0.413
Relative Frequency of 1: 192/1000 = 0.192
Relative Frequency of 2: 157/1000 = 0.157
Relative Frequency of 3: 116/1000 = 0.116
Relative Frequency of 4: 62/1000 = 0.062
Relative Frequency of 5: 31/1000 = 0.031
Relative Frequency of 6: 14/1000 = 0.014
Relative Frequency of 7: 8/1000 = 0.006
Relative Frequency of 8: 3/1000 = 0.003
Relative Frequency of 9: 3/1000 = 0.003
Relative Frequency of 14: 1/1000 = 0.001

2. **Sequences and Angles**

Case: s=6 r=1
Random Sequence:
['', 'c2', 'c2c3', 'c3c1c2', 'c1c2c1c2', '',
'', '']
Random Angle:

['0.809025655125pi', '0.56855869725pi', '0.540848160144pi',
'0.204319881203pi', '0.870065010651pi', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '']
Systematic Sequence:
['', 'c1', 'c1c2', 'c1c2c1', 'c2c3c2c3', '',
'', '']
Systematic Angle:
['0.999493042617pi', '0.957514334827pi', '0.960512813955pi',
'0.96251180004pi', '0.536727763885pi', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '']

Case: s=6 r=2

Random Sequence:
['', 'c3', 'c2c1', 'c1c2c1', 'c3c1c2c1', 'c3c2c3c2c3', 'c2c1c2c1c2c1',
'c2c1c2c1c2c3c1', 'c1c2c1c2c1c2c1c2', 'c2c3c2c3c2c3c2c3c2', '', '', '', '', '', '', '',
'', '']
Random Angle:
['0.736255612043pi', '0.223218463911pi', '0.631622553266pi',
'0.984401772169pi', '0.159624977709pi', '0.336351245895pi',
'0.668433585353pi', '0.662340559842pi', '0.832920335197pi',
'0.500220780209pi', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '']
Systematic Sequence:
['', 'c1', 'c1c2', 'c1c2c1', 'c1c2c1c2', 'c1c2c1c2c1', 'c1c2c1c2c1c2',
'c1c2c1c2c1c2c3', 'c1c2c1c2c1c2c1c2', 'c1c2c1c2c1c2c1c2c1', '', '', '', '',
'c2c1c2c1c2c1c2c1c2c1c2c1', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '']
Systematic Angle:
['0.818584801903pi', '0.948518897444pi', '0.97750419568pi',
'0.989498112191pi', '0.994495577404pi', '0.997494056532pi',
'0.998493549574pi', '0.835576183628pi', '0.999493042617pi',
'0.833577197543pi', '', '', '', '', '0.666661859426pi', '', '', '', '', '', '', '', '', '', '', '', '',
'', '']

Follow Up:
Initially, before any coding implementation, I suspected that the longest orbit would be from a system that consisted of spheres with r = 2 and s=6 since this would maximize the potential for the particle to reflect as many times as possible. As for which angle generating process would create the longest orbit initially did not occur to me. However I realized upon implementing my systematic angle generator that for a large number of trials, I would increment the angle equally depending on however many trials I ran. By doing this, I essentially simulate a full 360 degrees with small gaps in between. This, I then suspected, would give me potentially more hits by testing more consistently and systematically.