Computational Physics

FINAL PROJECT REPORT

Interactive Galton Board Simulation:

Observation and Comparison Analysis

Prepared by:

Zhongli's Abandoned Children

(Daniel, Filbert, Jocelin & Tiffany)

Even semester 2nd year 2023

Computer Science Program

Binus University International

Course Name: Computational Physics
Class : L4AC

Lecturer: Dr. Maria Seraphina Astriani, S.Kom., M.T.I

Type of Assignments: Final Project

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

Binus International seriously regards all forms of plagiarism, cheating, and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity, and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept, and consent to Binus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

1. Benedictus Filbert Federico
2. Daniel Eric Phangandy
3. Jocelin Wilson
4. Tiffany Widjaja

Table of Contents

I. Introduction

a) Background

The Galton Board is a device used to demonstrate the principles of probability and statistics. It was first invented in the 19th century by an English mathematician and scientist named Sir Francis Galton. The Galton Board consists of a vertical board with a series of evenly spaced pegs or pins arranged in a triangular pattern. At the top of the board lies a funnel-shaped structure through which a number of small balls are dropped. As the balls fall, they bounce off the pegs and randomly deviate either to the left or the right, following a path dictated by the pegs. At the bottom of the Galton board, there are a series of evenly spaced slots. These slots collect the balls as they exit the board. The number of slots corresponds to the number of pegs in the board.

The Galton board demonstrates the principles of the binomial distribution, which describes the probability of different outcomes when there are only two possibilities of the ball bouncing either right or left. As the balls fall and bounce off the pegs, they accumulate in the slots below. The pattern formed by the balls in the bins resembles a bell-shaped curve, known as the normal distribution or Gaussian distribution. It is a visual representation of how random events can lead to predictable patterns over time. It shows that even though individual ball paths may seem unpredictable, when many balls are dropped, their collective distribution follows a consistent pattern. This is a fundamental concept in statistics and probability theory.

b) Problem

Recently, records are pointing towards poor performance shown by students in mathematics at secondary school level. Consequently, more efforts are needed to reduce this rate of failure and low interest in mathematics in public examinations. A study reveals that students' poor academic achievement in mathematics is partly due to the selected method of teaching used (Ishak et al. 2019). This is an issue since (Khan and Salman. 2020) according to another research, mathematics is the key to success in every field and is always used daily in our routines. For instance, mathematics is required to get relevant jobs and to be part of the market of success. The study finds that the use of games and simulation environments led to improved achievements and positive attitudes towards mathematics (Ishak et al. 2019). It concluded that teachers' use of interactive teaching methods would go a long way in sustaining and motivating students' interest in learning Mathematics. The study recommended that teachers should be encouraged to use games and simulations in teaching Mathematics in secondary schools. d

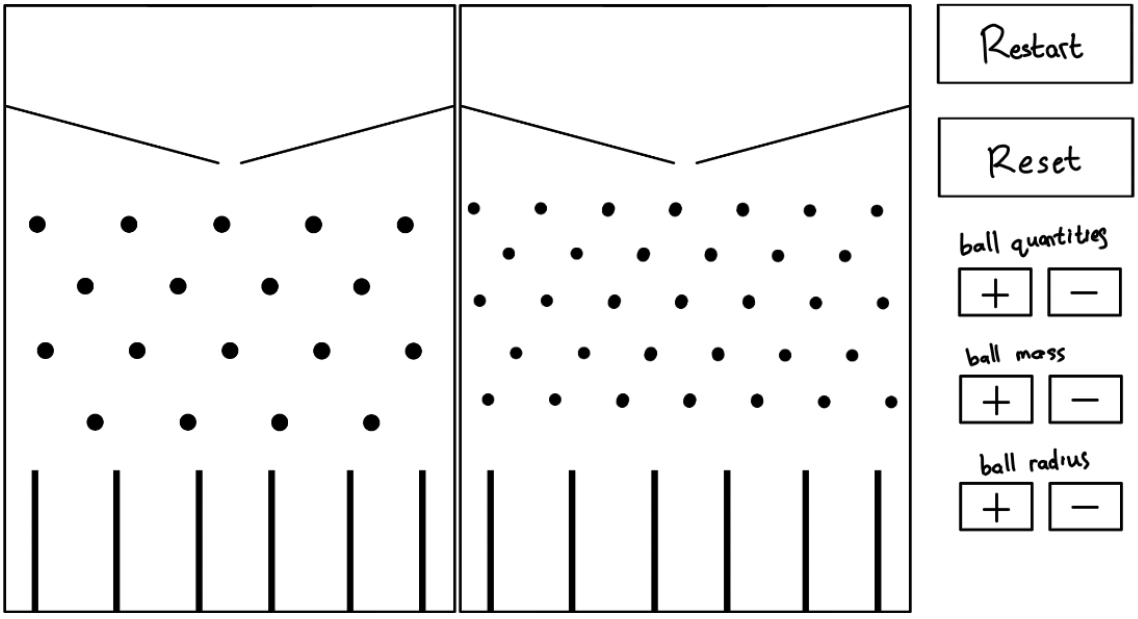
c) Proposed Solution

Hence why we proposed to create an Interactive Galton Board Simulation to support students' learning experience during their mathematics classes. The Galton board can be used as an educational tool during their learning journey to help understand more complex concepts such as probability, randomness, and the central limit theorem. It provides them a fun and interactive way to explore statistical concepts and observe how patterns emerge from randomness. The purpose of making it interactive is to allow students or teachers to experiment with different sizes and mass of the balls when simulating the galton board. It helps them observe how those slight changes may affect the result of the distribution pattern. With this proposed method, students and teachers are able to analyze and make observations themselves or to assist their future research in class. The main goal of this simulation is hopefully to contribute to increasing students' performance as well as their attitude towards mathematics. In addition, we not only propose one

galton board but two galton board simulations that are different in the amount of slots and nails in the program to enable comparison analysis studies of the distribution results.

d) Features

This program includes two different galton board simulations that can run at the same time upon running the file. It contains a reset button which allows users to reset the simulation by running the program from the start with the balls spawned from the top of the board through the funnel. This reset button will also reset the radius, mass and quantities of the balls to its default or initial state. Furthermore, the program also contains a restart button. This restart button allows users to rerun the program with the same settings the user has set to the balls before. The function of this is to allow users to re-experiment with the same settings multiple times to record the data of the distribution results. Moreover, the program also contains features for users to change the radius, mass and the quantity of the balls through the UI text box buttons provided. It will include a + and - text button icon for the user to increase or decrease the respective radius, mass or quantity of balls in the program. Below is a draft UI of the interactive galton board simulation program with it's intended features:



e) Job divisions:

These are the roles assigned to each of our team members.

- Main coder: Filbert
- Assist coder: Tiffany
- Main Author: Tiffany
- Co Authors: Daniel & Jocelin & Filbert
- UI designer + Idea: Jocelin & Tiffany
- PPT Slides Designer: Jocelin
- Video Demo Editor: Filbert

A main coder's job is to responsibly handle the creation of the program. It is their duty to implement the physics properties and UI design that the team has prepared. The job of the assist coder is to contribute in creating the code along with the main coder and help debug any problems found along the way. As for the UI designer, their job is to come up with the idea of the program itself and design the features and how they want the program to look like. The main author is in charge of mainly writing and proofreading the whole report before it is submitted. It is also their responsibility to ensure that all the contents necessary in the report are included. The co author is in charge of helping write sections of the report and creating citations for it. PPT slides designer's responsibility is to create the slides including inputting the required contents that has gained the team's approval to include. Lastly, the job of the video editor is to document the program with a recording software of any choice and edit it.

II. Related Works

A research titled "Creating kappa-like distributions from a Galton board" by M. Leitner, M.P. Leubner, and Z. Vörös explores the concept of Galton board from the perspective of long-range interactions and nonextensive statistics. This investigates the corresponding power-law distribution known as kappa-distribution which is a consequence of entropy generalization. The research demonstrates that the Galton board can reproduce both positive and negative kappa-like distributions. The findings suggest a transition from normal distributions to kappa-like distributions with positive kappa-like exponents associated with memory effects and negative kappa-like exponents to enhanced interactions.

A research titled "Improving mixture of grains by using bi-dimensional Galton boards" by J.G. Benito, I. Ippolito, and A.M. Vidales presents an experimental and numerical study that deals with the problem of mixing grains falling down through a bi-dimensional Galton board (BGB). It examines the influence of lateral walls within the BGB on the mixing process where they release disks of different species from the top of the board and interact with obstacles and the lateral walls during their descent. Then they analyze the distribution of particles at the bottom of the board and the impact of the presence of walls. The findings suggest that the presence of walls significantly improves the quality of the particle mixture.

A research titled "Galton Board" by Valerii V. Kozlov and M. Yu. Mitrofanova presents results of simulations of the Galton board with various degrees of elasticity of the ball-to-nail collision by using an interactive Microsoft Visual C++ application. The experiments are conducted by varying the parameters of the problem, such as the coefficient of restitution, the nail's radius, and the variance of the initial distribution of the balls. The results are analyzed by generating histograms of the balls' distribution over the compartments and calculating the variance of the final distribution. Results show that when the objects collide with high coefficient of restitution, the distribution of balls in compartments is nearly Gaussian with a few noticeable pits. As the coefficient of restitution decreases, the normal distribution gets corrupted which results in distinctive gaps to appear. Below this point, the gaps disappear, and the distribution becomes similar to a Gaussian pattern. The nail's radius also affects the distribution, causing deviations from a Gaussian shape.

In comparison to the previous research, our research expands upon the concept of Galton boards by investigating the adjustable parameters of ball size, quantities, and mass similar to the "Galton Board" research with elasticity and nail radius. By introducing this flexibility and customizability, we aim to explore the effects of these adjustable parameters within the Galton board simulation. Through our experimental and numerical approaches, we aim to contribute to the understanding of Galton boards with our customizability to give users control over the independent variables.

III. Implementation

a) Physics Formula

Galton Board

P left = 1- p right

The probability of the ball going to the left is 1 minus the probability of a ball going right. If the pegs are symmetrically arranged, probability of left = probability of right = ½.

Pascal's Triangle Path Probability

0.5number of rows - 1

We subtract 1 from the row count since there is no choice involved on the first row, there only being one number.

Binomial Distribution

$P(r) = nCr \cdot pr\ (1 - p)n-r$

Gravitational Acceleration Constant

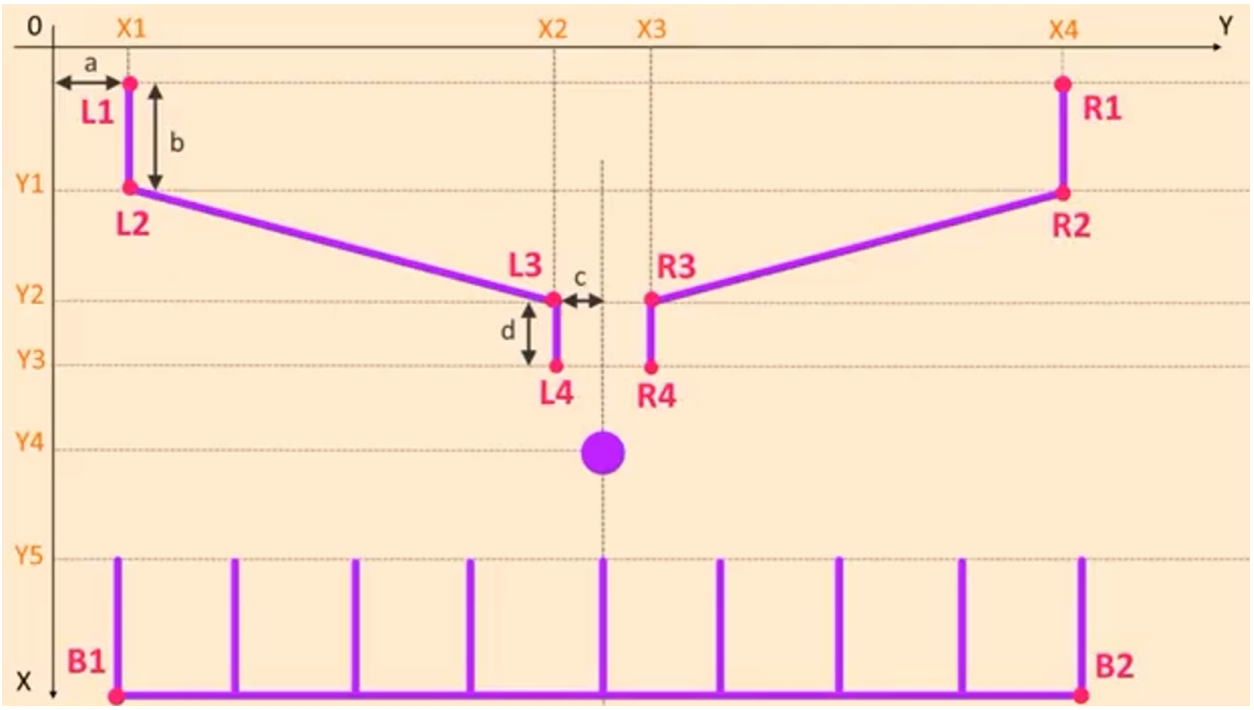
g= 9.81m/s^2

The gravity of earth, denoted by g.

Note: Not all formulas mentioned are applied into the simulation program. For instance, the binomial distribution formula is only mentioned, since that is the result of the distribution the galton board created itself.

b) Method to Apply Formula (in Python)

In the creation of the galton board, we use pygame and pymunk. These libraries require coordinates to draw in any figures we desire. The image below is how we label the junctions for pymunk to draw. Using coordinates such as L1 - L4 draw the left side of the funnel while R1-R4 draw the right side. As seen in the image, declared variables such as a, b , c, and d are used for the length of the funnel which corresponds to how far the distance is. B1 and B2 are the coordinates for the base or the floor to capture the balls and prevent them from falling off the map.



a, b, c, and d is a predetermined default value, using those values as the formula to create other variables. To create 2 galton boards, a wall separator is needed to isolate each one of the galton boards to remove any external influence. The separator is called by a variable: split.

```
#galton funnel
a, b, c, d = 10, 100, 18, 40
x1, x2, x3, x4 = a, split // 2 - c, split // 2 + c, split - a
y1, y2, y3, y4, y5 = b, height // 4 - d, height // 4, height // 2 - 1.5 * b, height - 4 * b
L1, L2, L3, L4 = (x1, -100), (x1, y1), (x2, y2), (x2, y3)
R1, R2, R3, R4 = (x4, -100), (x4, y1), (x3, y2), (x3, y3)
B1, B2 = (0, height), (split, height)
```

The function below is to create the nails and walls for the slots. The iteration consists of 2 loops or 2 rows that consistently loops according to the y range set in the for i in range(). For i loop create the second row while for j loop creates the first row. To create a checkered grid the second row is 0.5 unit to the right while -1.5 to the left for the first row. A nested if creates the slot walls,

```
#pegs & walls
peg_y, step = y4, 68
#                  ^ y intervals
for i in range(10):
    #           ^ range of y axis
    peg_x = -1.5 * step if i % 2 else -step
    for j in range(split // step + 2):
        #           ^ range for x
        create_peg(peg_x, peg_y, space, 'darkslateblue')
        if i == 8:
            #create wall
            create_segment((peg_x, peg_y + 80), (peg_x, height), segment_thickness, space, 'darkslategray')
        peg_x += step
    peg_y += 0.5 * step
```

it follows the first row x axis coordinates on where to put down the walls, these coordination are required to create the environment for the galton board

```
#ball floor, the funnel #1
platforms1 = (L1, L2), (L2, L3), (L3, L4), (R1, R2), (R2, R3), (R3, R4)
for platform1 in platforms1:
    create_segment(*platform1, segment_thickness, space, 'darkolivegreen')
    #floor
create_segment(B1, B2, 20, space, 'darkslategray')

#split wall
create_segment(S1, S2, 10, space, 'black')
create_segment(S11, S22, 10, space, 'black')

# ball floor, the funnel #2
platforms2 = (L11, L22), (L22, L33), (L33, L44), (R11, R22), (R22, R33), (R33, R44)
for platform2 in platforms2:
    create_segment(*platform2, segment_thickness, space, 'darkolivegreen')
    #floor
create_segment(B11, B22, 20, space, 'darkslategray')
```

The code above creates the funnel, split wall, floor from the declared variables. The function take the coordinates from the list of variables into create_segment(). The create_segment() requires coord, thickness, space, and color as the parameters

Now each ball belonging to the galton board is created by a function called create_ball() as seen in the snippet of the code below. The physics of the ball is defined within this function such as the ball's moment, body, mass, elasticity, friction, radius and more.

```
#create ball
def create_ball(space):
    moment = pymunk.moment_for_circle(ball_mass, 0, ball_radius)
    body = pymunk.Body(ball_mass, moment)
    body.position = randrange(x1, x4), randrange(-y1, y1)
    shape = pymunk.Circle(body, ball_radius)
    shape.elasticity = 0.1
    shape.friction = 0.1
    space.add(body, shape)
    return body, shape
```

Each ball must have a moment of inertia and this is defined in the ball_moment = pymunk.moment_for_circle(ball_mass, 0, ball_radius). The moment of inertia is the quantitative measure of the rotational inertia of the body such as the opposition that the body exhibits to having its speed rotation about an axis altered by the application of a torque. The balls also have a friction attribute and so we need to set a coefficient of friction for objects in ball_shape.friction. This is so that the balls will make a certain turn upon impact to make the simulation look more realistic.

```
#initialize pymunk space and set gravity
space = pymunk.Space()
space.gravity = 0, 9810
```

For the gravity, our program has attempted to implement the same numbers which is 9.81 however using 9.81 causes the balls to fall down really slow. This is because the unit of gravity is in m/s^2 meanwhile with pymunk, there is no set unit and most programs can freely decide the number of gravity as anything. To simulate the gravity, the number 9.81 is multiplied by 1000 resulting in the number 9810.

To start the simulation we need to spawn in the balls, initially upon running, the program will pre-spawn the balls using a loop. Utilizing a list to hold spawned balls for later use such as deleting the balls upon calling reset() or restart(). The for loop calls a certain number of balls regulated by the num_balls variable.

```
#list to hold balls
balls = []
#pre-spawn balls
for _ in range(num_balls):
    ball = create_ball(space)
    balls.append(ball)
```

Deleting the balls from the list requires another loop. Using the remove method, taking from the balls list which was appended from the balls creation

```
def restart_balls():
    global balls
    for ball in balls:
        space.remove(ball[0], ball[1])
```

c) Implementation Details

Flow of the Program

```
#                x,y coords x,y size  text on button
button = Button(1520, 20, 170, 100, "Restart", restart)
button1 = Button(1520, 150, 170, 100, "Reset", reset)
button2 = Button(1520, 310, 70, 50, "+", increase_balls)
button3 = Button(1600, 310, 70, 50, "-", decrease_balls)
button4 = Button(1520, 420, 70, 50, "+", increase_mass)
button5 = Button(1600, 420, 70, 50, "-", decrease_mass)
button6 = Button(1520, 535, 70, 50, "+", increase_radius)
button7 = Button(1600, 535, 70, 50, "-", decrease_radius)
```

The above image shows how the buttons are created in the code and which functions gets called upon the user clicking it.

Input: press button "restart"
Process: python recognise this as an event and execute restart(). This restart function will also execute restart_balls() and restart_balls2()
Output: The program will run again from the beginning meaning the balls will respawn back from the top with the previous settings the user has implemented using the UI text buttons.

```
def restart():
    restart_balls()
    restart_balls2()
```

Input: press button "+" for ball quantities
Process: the increase_balls() function is executed which calls out the restart_balls() function for both boards.
Output: This will also cause the program to rerun upon clicking the + button however it will also increase the amount of balls spawned by 50. For instance if we start with 350 balls and upon clicking the "+", it will run with 400 balls instead.

Input: press button "-" for ball quantities
Process: the decrease_balls() function is executed which calls out the restart_balls() function for both boards.
Output: This will also cause the program to rerun upon clicking the - button

```
#edit ball quantity
def increase_balls():
    global num_balls
    num_balls += 50
    restart_balls()
    restart_balls2()
    print(f"num_ball(+):{num_balls}")

def decrease_balls():
    global num_balls
    if num_balls > 50:
        num_balls -= 50
        restart_balls()
        restart_balls2()
    print(f"num_ball(-):{num_balls}")
```

however it will also decrease the amount of balls spawned by 50. For instance if we start with 350 balls and upon clicking the "-", it will run with 300 balls instead.
Input: press button "Reset"
Process: python also recognise this as an event and executes reset() upon the click. As seen from the image below, it removes the balls and appends ball into the empty array called balls.
Output: The program will run again from the start where the balls will respawn back from the top however all previous settings gets reset back to its default initial settings that has been determined in the program code. Print all the variables in the terminal

```
def reset():
    global balls, num_balls, ball_radius, ball_radius2, ball_mass, ball_mass2
    num_balls = 350
    ball_radius = 7
    ball_radius2 = 7
    ball_mass = 1
    ball_mass2 = 1
    for ball in balls:
        space.remove(ball[0], ball[1])
    balls = []
    for _ in range(num_balls):
        ball = create_ball2(space)
        balls.append(ball)

    global balls2
    for ball in balls2:
        space.remove(ball[0], ball[1])
    balls2 = []
    for _ in range(num_balls):
        ball = create_ball(space)
        balls2.append(ball)
    print("reset")
    print(f"num_ball:{num_balls}")
    print(f"ball_mass:{ball_mass}")
    print(f"ball_radius:{ball_radius}")
```

Input: press button "+" for ball mass
Process: the increase_mass() function is executed which calls out the restart() function.
Output: This will also cause the program to rerun upon clicking the + button however it will also increase the mass of all balls spawned by 5. For instance if we start with the mass of the balls set as 1 and upon clicking the "+", all of the balls will have a mass of 6.

Input: press button "-" for ball mass
Process: the decrease_mass() function is executed which calls out the restart() function.
Output: This will also instruct the program to rerun upon clicking the - button however it will also decrease the the mass of all balls spawned by 5. For instance if we start with the mass of the

balls set as 10 and upon clicking the “-”, all of the balls will end up having a mass of 5. Print ball_mass with plus(+) or minus (-) based on what the input

```
#edit mass
def increase_mass():
    global ball_mass, ball_mass2
    ball_mass += 5
    ball_mass2 += 5
    restart()
    print(f"ball_mass(+):{ball_mass}")

def decrease_mass():
    global ball_mass, ball_mass2
    if ball_mass > 6 and ball_mass2 > 6:
        ball_mass -= 5
        ball_mass2 -= 5
    restart()
    print(f"ball_mass(-):{ball_mass}")
```

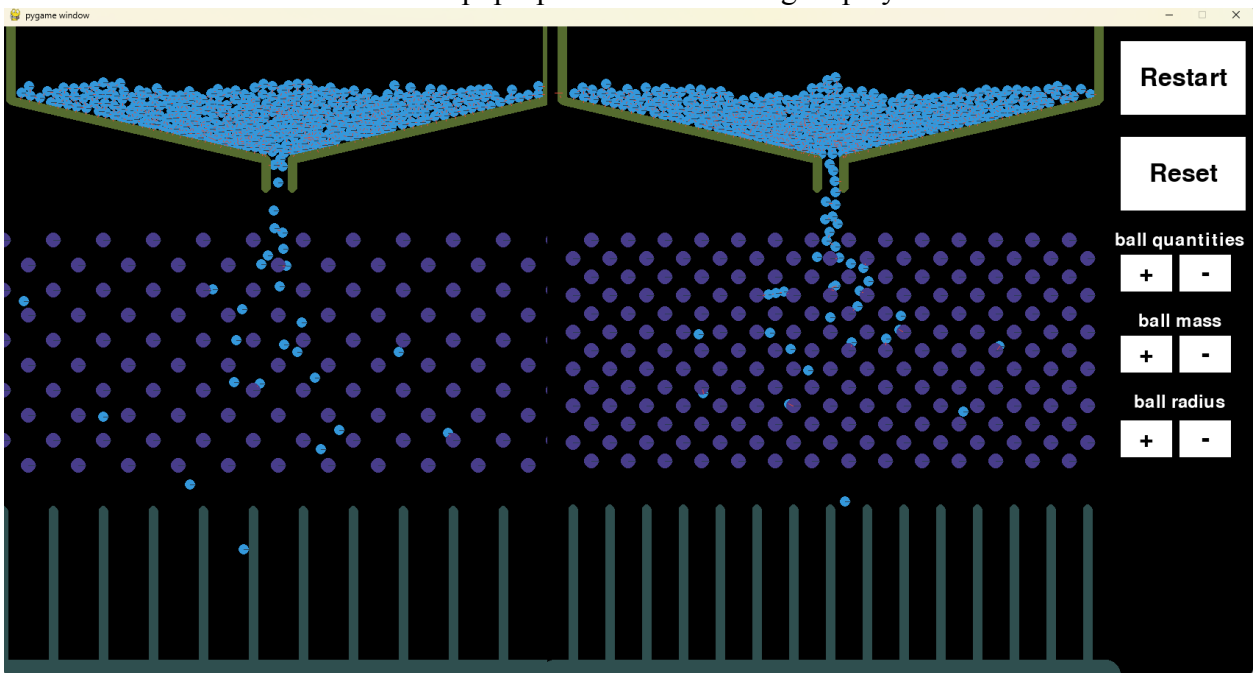
Input: press button “+” for ball radius
Process: the increase_radius() function is executed which calls out the restart() function for both boards as well.
Output: This will also cause the program to rerun upon clicking the + button however it will also increase the radius of all balls spawned by 0.25. For instance if we start with the radius of the balls set as 7, upon clicking the “+”, all of the balls will have a radius of 7.25.

Input: press button “-” for ball radius
Process: the decrease_radius() function is executed which calls out the restart() function for both boards as well.
Output: This will also cause the program to rerun upon clicking the - button however it will also decrease the radius of all balls spawned by 0.25. For instance if we start with the radius of the balls set as 8.25, upon clicking the “-”, all of the balls will have a radius of 8.

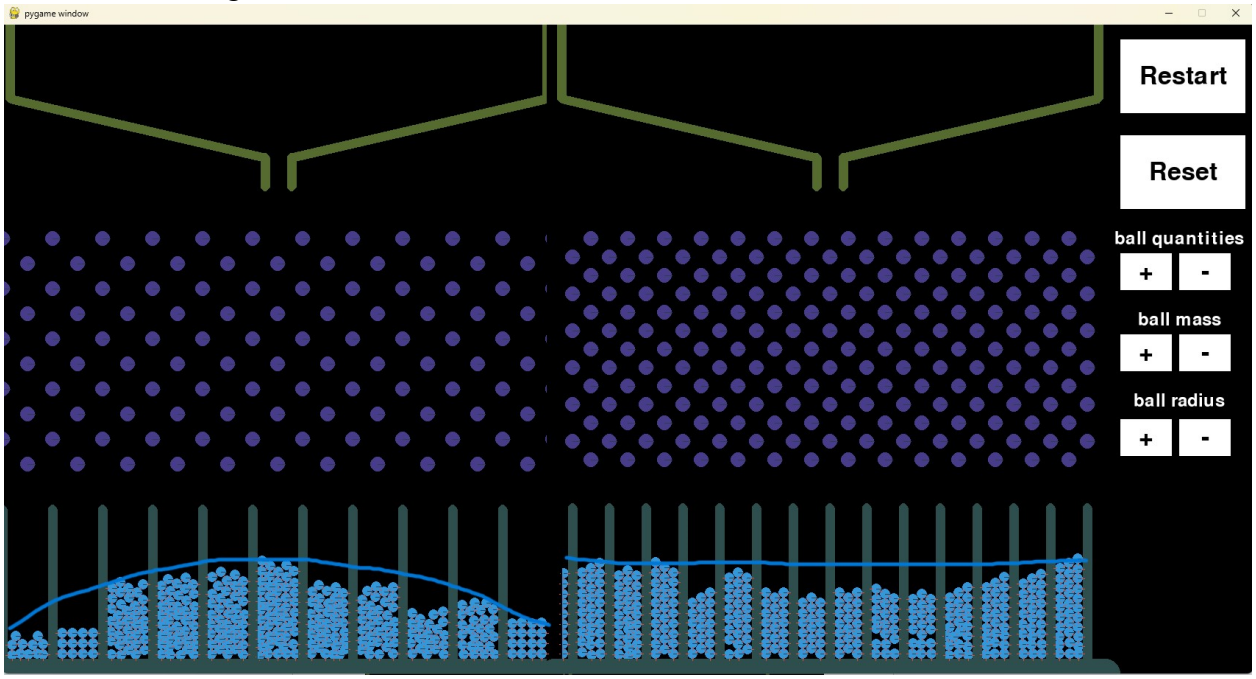
```
#edit radius
def increase_radius():
    global ball_radius, ball_radius2
    ball_radius += 0.25
    ball_radius2 += 0.25
    restart()
    print(f"ball_radius(+):{ball_radius}")

def decrease_radius():
    global ball_radius, ball_radius2
    if ball_radius > 0 and ball_radius2 > 0:
        ball_radius -= 0.25
        ball_radius2 -= 0.25
    restart()
    print(f"ball_radius(-):{ball_radius}")
```

Result of the Program:

Once the code is run a window will pop-up with the following display:



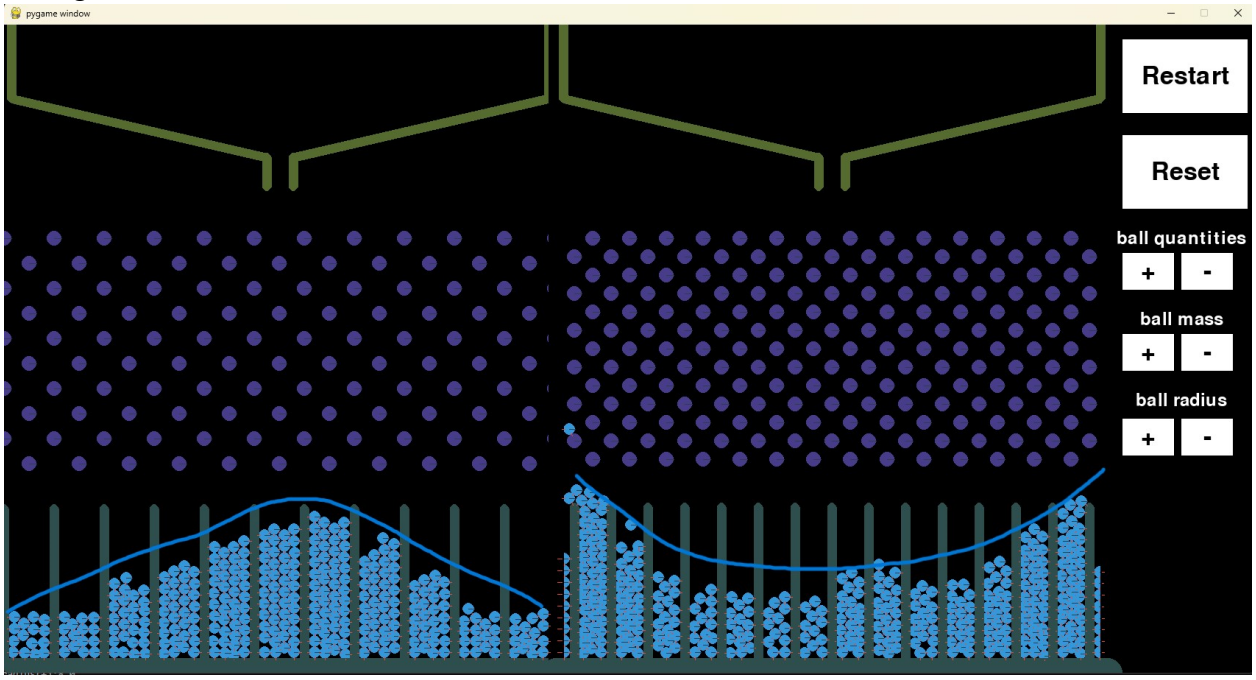
On the left, the pegs are more spread apart whereas the pegs are more tightly packed on the right side. After running the simulation, the end result is as follows:



As seen from the above screenshot, it can be concluded that the side with more spread out pegs will create a bell curve, which is the binomial distribution that approximates a normal

distribution as a regular Galton Board would generate. This contrasts with the result on the right which shows a flatter distribution among the balls.

After further testing the results remained the same when increasing and decreasing the ball quantity and mass, however the results differed when the radius of the balls were increased. Giving a different result



As seen in the above screenshot the result in the left side remains the same, showing another binomial distribution, however with more balls lending on the left side rather than on the right. However, the right side shows a significant change in the results. It now shows the complete opposite distribution to the binomial distribution, which is known as the bimodal distribution.

d) Tests

We decided to conduct several tests on our program using different settings of the ball mass, ball radius, and amount of balls spawned. On our first test, we will use the default settings where the ball mass is set to 1, the ball radius set to 7, and ball quantities are set to 350. This can be seen in our program py file:
num_ball:350
ball_mass:1
ball_radius:7

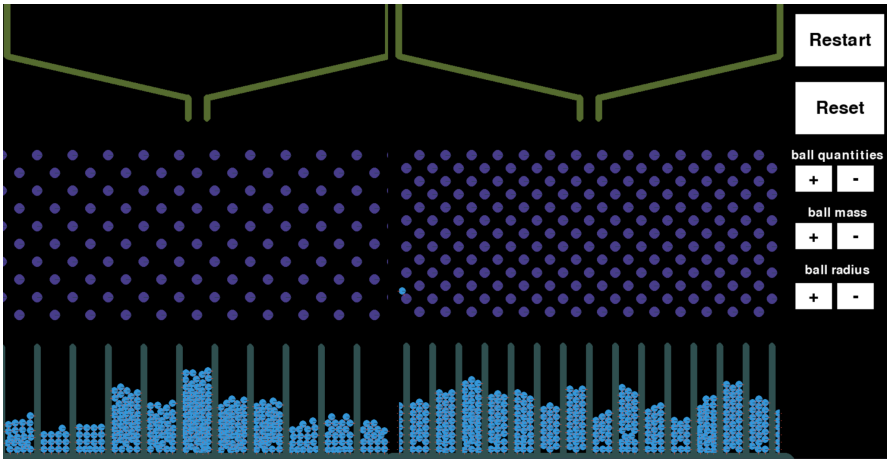
Our second test uses these settings:
num_ball: 350
ball_mass: 1
ball_radius: 8.25

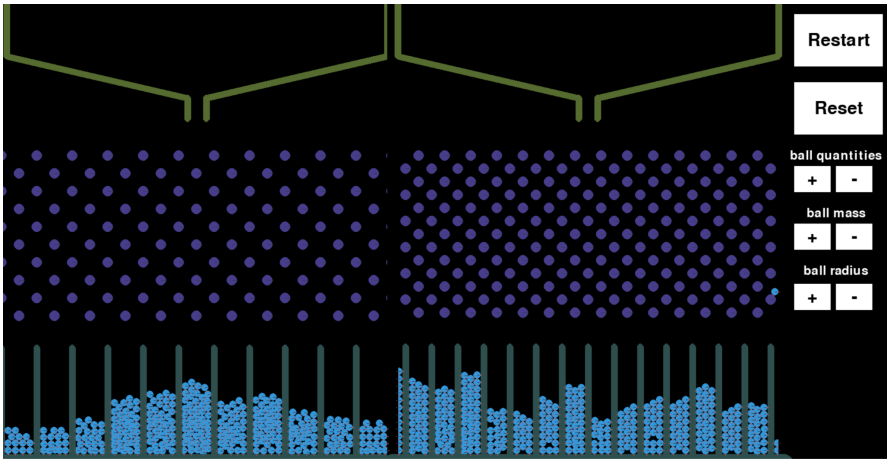
The third test uses these settings:
num_ball: 650
ball_mass: 1

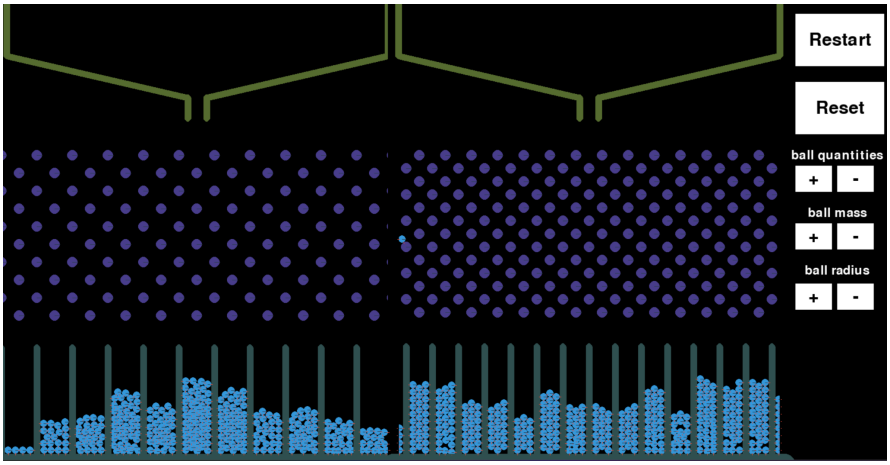
ball_radius: 6

The fourth test uses these settings:
num_ball: 650
ball_mass: 500
ball_radius: 7
The results of the tests will be displayed below:

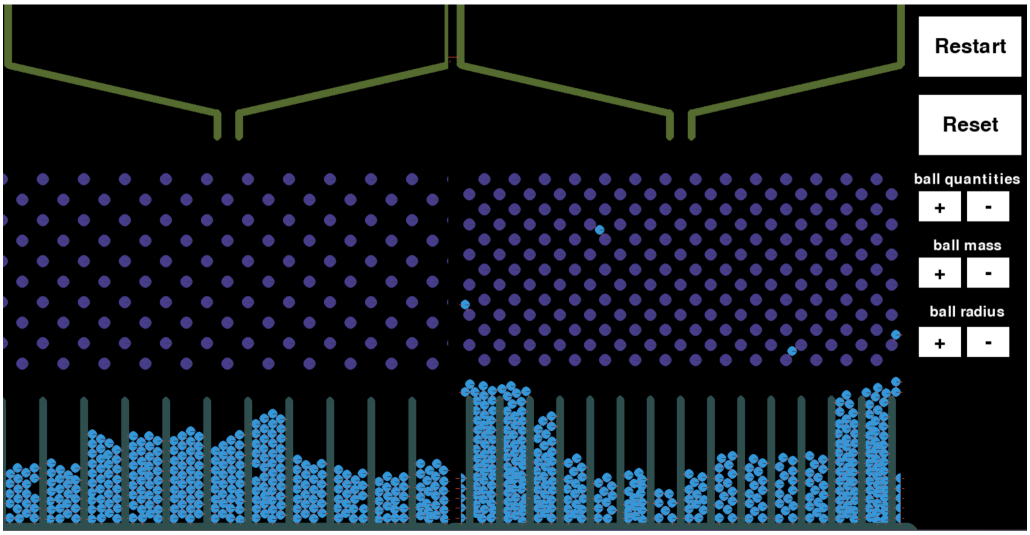
First Test results:

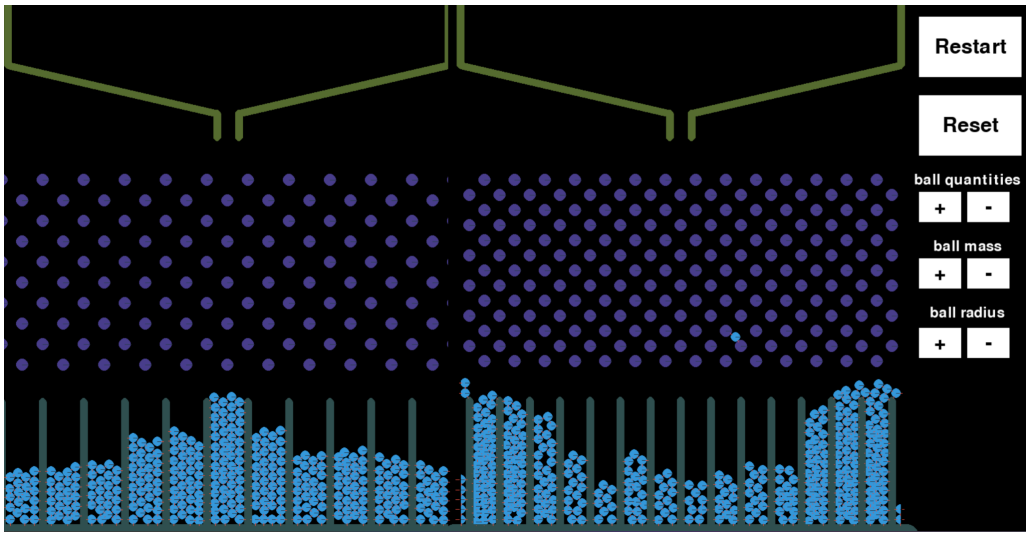


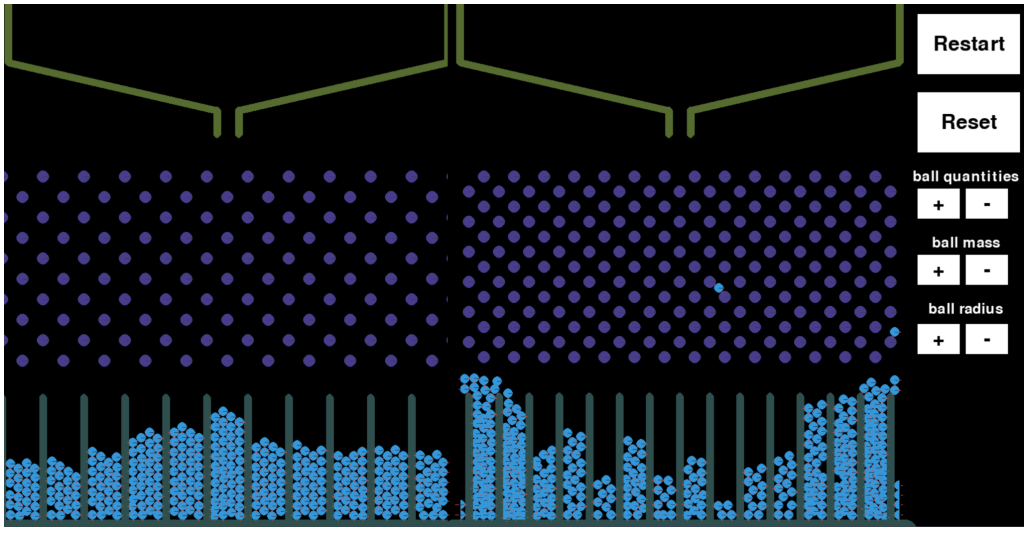




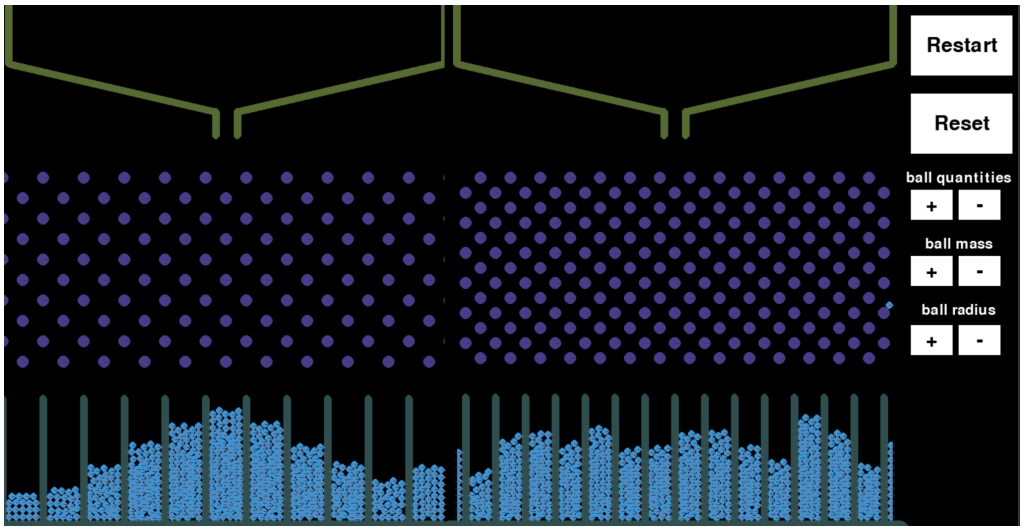
Second Test results:

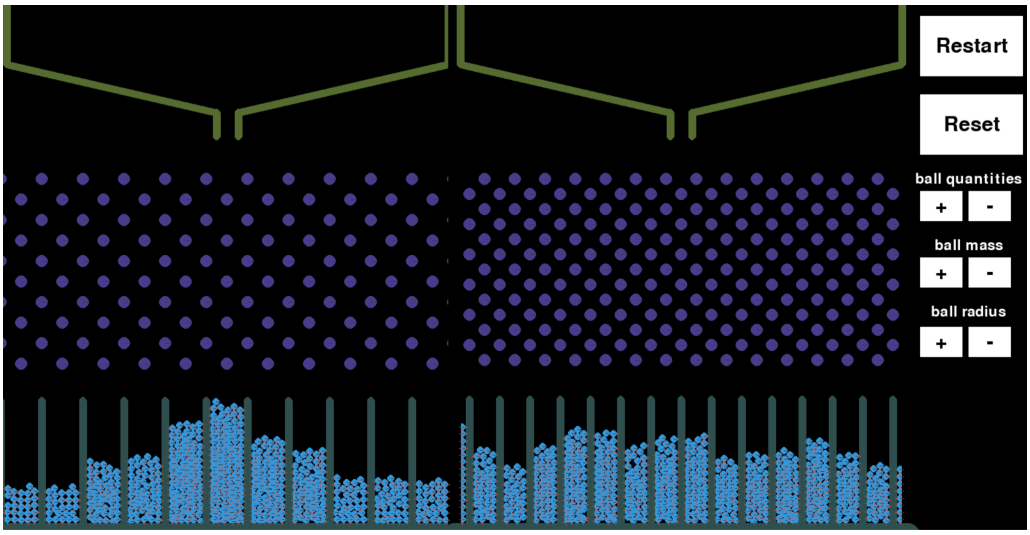


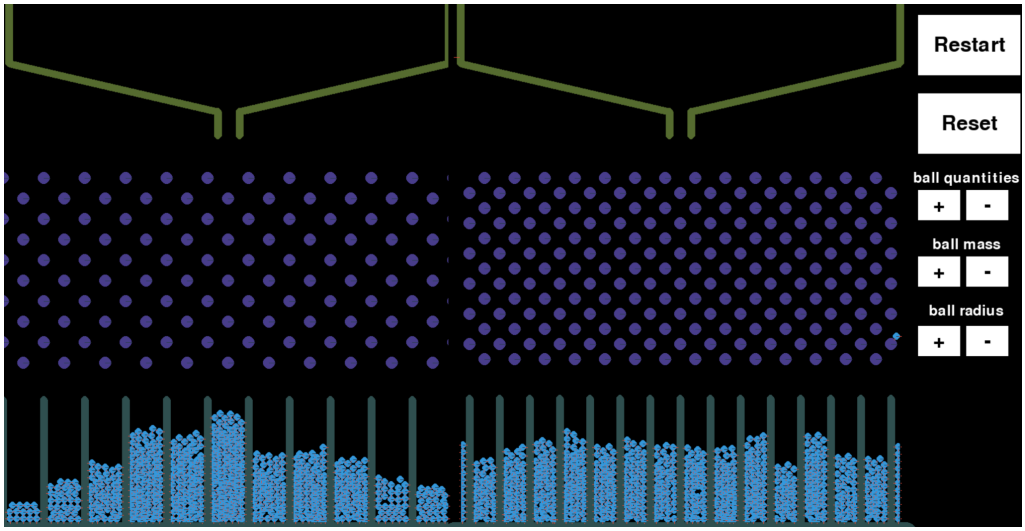




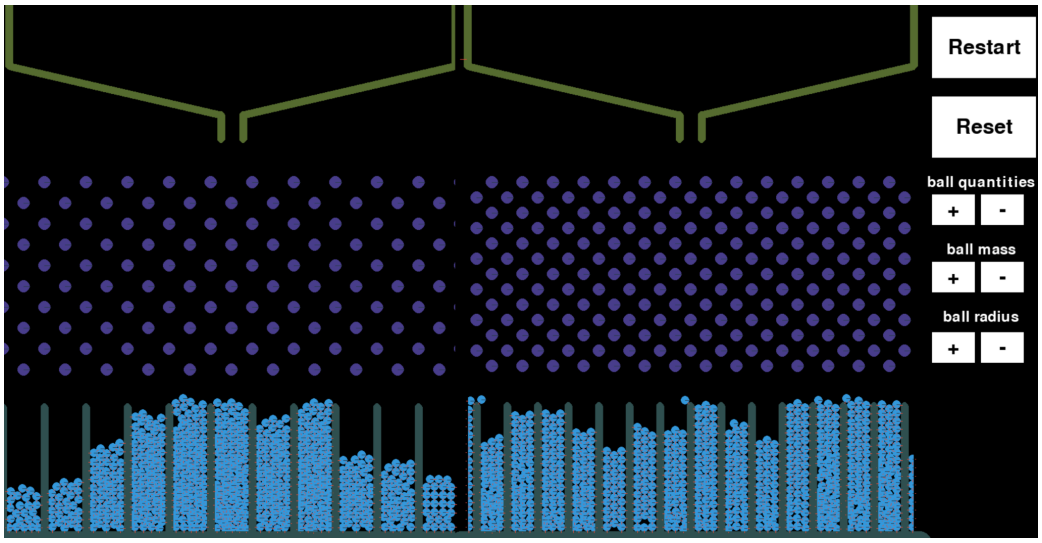
Third Test results:

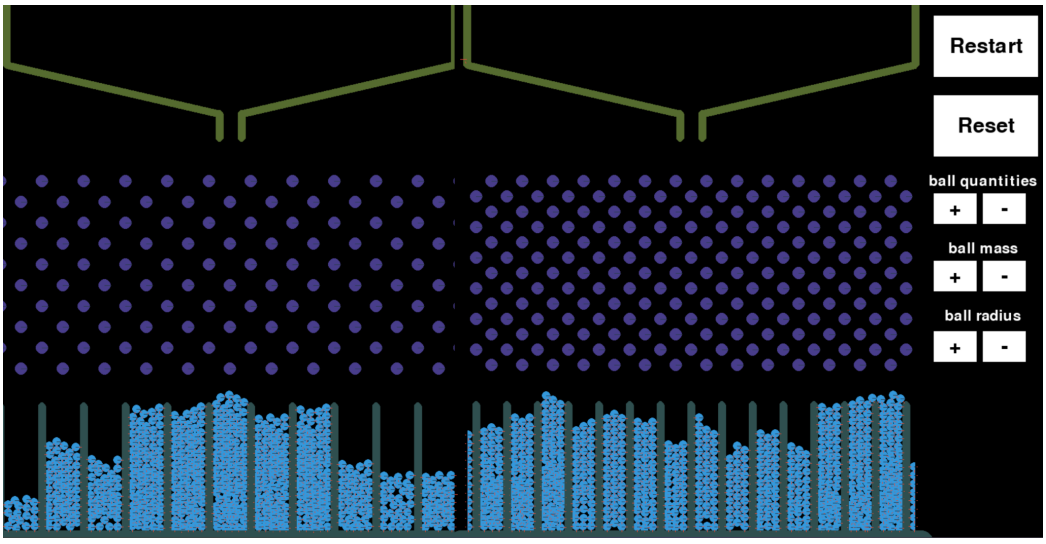


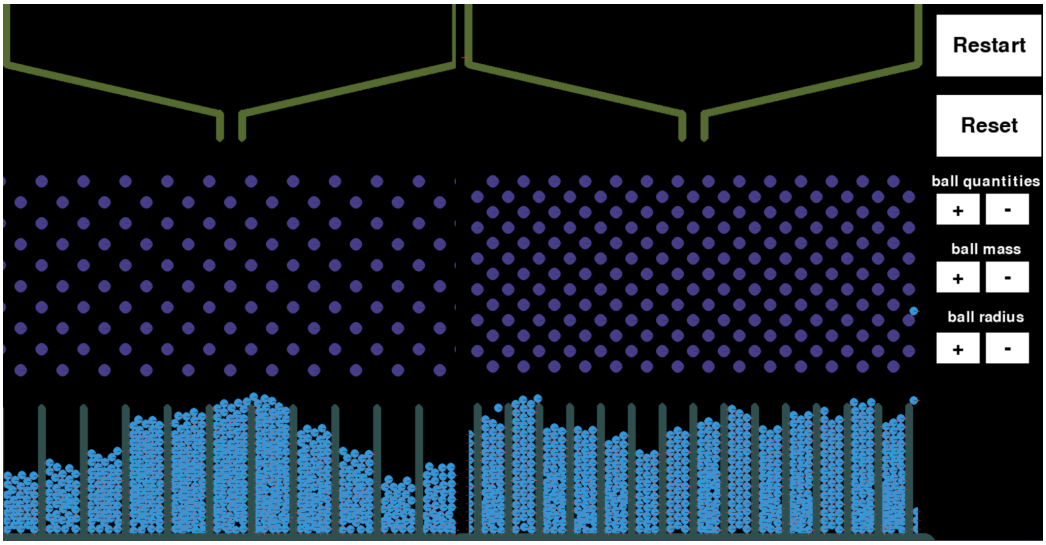




Fourth Test results:







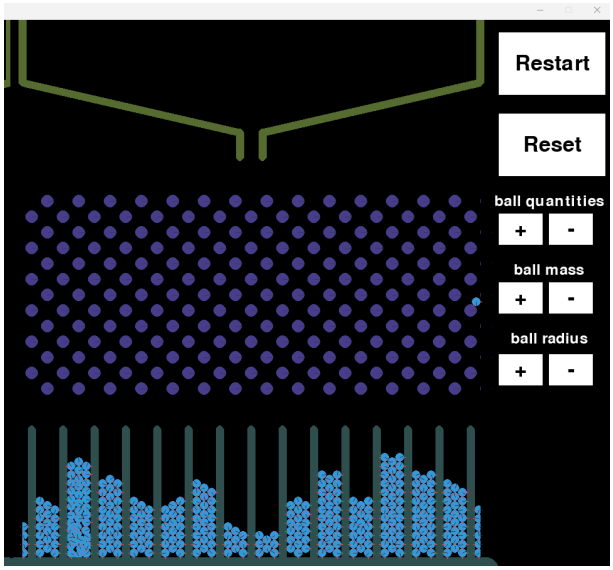
IV. Evaluation & Discussion

a) Proof of Problem solved

In a technical program perspective, we have successfully implemented an interactive galton board simulation for others to perform tests on. This is because the left galton board forms a normal distribution or binomial distribution which is the intended outcome of what a real life galton board should simulate. On the other hand, the right side forms a different distribution than a normal distribution, possibly the multimodal distribution, which is suitable for further analysis for others to conduct. Furthermore, the buttons have also been successfully implemented and are working as intended. The amount of balls, its mass and radius can be altered by the user through the provided UI buttons. From several tests with different settings, the program has enabled users to observe how these various settings affect the result of the distribution. This means our end result has managed to solve one of the objectives which is to help users conduct observations on probability and distribution with the help of our tool. However, talking in a global perspective, we have yet managed to see the outcome or the scale of impact it brings from our tool alone such as increasing the performance of students in mathematics. What has been accomplished is, our team managed to create a simulation tool that will be useful for learning purposes in schools or for self learning at home. Our research believes that it will only serve as a real solution to the current problem once our program has been distributed to the public for others to access and utilize.

b) Analyze the Tests

From the first test results, it is already visible enough to see the normal distribution pattern forming in the left side of the galton board. However, the right side of the galton board forms an unknown distribution. The first test has shown that it is not recommended to use the default settings since the results for both distributions are quite inconsistent and this leads to more tests being conducted. The theory we propose after conducting the first test is, we assume that it may be trying to form a multimodal distribution which is a probability distribution with one mode for the galton board on the right side. To confirm this theory, we conducted our second test.



On the second test, where the radius of the balls are increased to 8.25, the left side of the Galton board forms the same distribution as the first test, demonstrating a pattern consistent with a normal distribution. However, it is on the right side where interesting findings emerge. Unlike the first test, the right side of the Galton board begins to exhibit a distinct and consistent distribution, confirming the presence of a multimodal distribution as predicted

earlier. This observation suggests that certain factors or variables are influencing the balls' trajectory, leading to the formation of multiple peaks in the distribution. Further testing will be conducted to explore the underlying causes and implications of this phenomenon, which will provide valuable insights into the behavior of the Galton board and its applications in probability and statistics.

On the third test, where the amount of balls are increased to 650, The left side shows a much more consistent distribution of the balls through the many tests conducted than the results of the first test. The right side shows a more flat distribution of balls as all of them land on the bins quite evenly. The reason for why the data seems to be more consistent on this test than the first is because of the higher population size or ball quantity, which shows that the more data that any given simulation, test or research has it will more often than not show more consistent and precise results.

On the fourth test, where the mass of the balls are increased to 500 and the radius is increased to 7, the left and right side show similar results to the first test that was conducted. This may be the case, however, upon further inspection of the results we can see that the amount of balls in the fourth test is seemingly larger than the test conducted in the third, which has the same quantity of 650 balls whereas the fourth test has the default quantity of 350. The theory crafted to explain this phenomenon is that in the simulation, the balls with a lower mass may have fallen out of the boundaries of the simulation whereas the balls with a higher mass managed to stay within the bounds of the simulation itself.

V. Conclusion & Recommendation

Our findings have shown that by experimenting with different settings of the balls quantities and radius, it can greatly change the results of the distributions created. For instance, the smaller the radius of the balls and the higher the ball quantities the program has, the clearer it is to see the binomial distribution formed by the left galton board. This is proven by the third test where the ball quantities are set to 650 with the radius set to 7 and mass as 1. As for the right side of the galton board, it will only form a bimodal distribution if the ball radius is set to 8 or 8.25 with the mass of 1 and ball quantities of 650. The difference in widths of the bins and pins has also contributed significantly in what kind of distribution the balls form as seen from our previous tests as it either results in a normal distribution in the left board or a bimodal one in the right board.

To further improve our research, it is recommended for others to further analyze the cause of forming the multimodal distribution or bimodal distribution. There are many simulations of the galton board that result in a normal distribution however only our findings have generated a bimodal distribution using the right side galton board. Currently there are very few bimodal distribution simulators hence it is suggested for others to perform more research on that field. It is also recommended for others to create a more interactive version of our simulator by allowing users to adjust the width of the bins and slots as well as the pins.

VI. References

Benito, J. G., Ippolito, I., & Vidales, A. M. (2008). *Improving mixture of grains by using bi-dimensional Galton boards.*https://doi.org/10.1016/j.physa.2008.05.003

Frost, J. (n.d.). *Bimodal Distribution: Definition, Examples & Analysis.* https://statisticsbyjim.com/basics/bimodal-distribution

Glenn, S. (n.d.). *Galton Board / Quincunx.* https://www.statisticshowto.com/galton-board/

Ishaq, A., Latunde, T., Mustapha, A., and Ogwumu, D. (2019). Impacts of Simulation-Games on Teaching and Learning Mathematics. *ResearchGate.* https://www.researchgate.net/publication/337427511_Impacts_of_Simulation-Games_on_Teaching_and_Learning_Mathematics

Khan, S., and Salman, R. (2020). Influence of Mathematics in Our Daily Lives. *MedCrave.* https://medcraveonline.com/AHOAJ/AHOAJ-04-00152.pdf

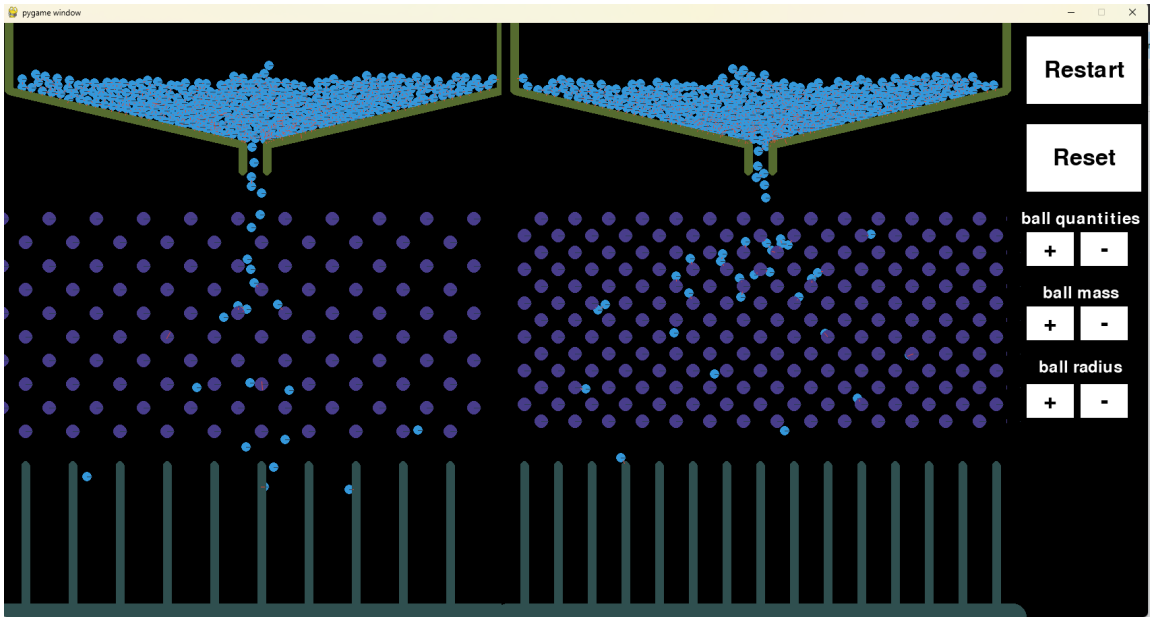
Kozlov, V. V., & Mitrofanova, M. Y. (2005). *Galton board*. https://arxiv.org/pdf/nlin/0503024

Leitner, M., Leubner, M. P., & Vörös, Z. (2011). *Creating kappa-like distributions from a Galton board.* https://doi.org/10.1016/j.physa.2010.11.044

VII. Appendices

a) Program Manual

1. In order to execute the program, Python, Pymunk and Pygame are a requirement for it to run. Please make sure you have all of those installed.
   Write this in your console:
   Pip install pymunk
   Pip install pygame
2. In an event where the program's window exceeds your display resolution: Make sure you go to display settings and change the scale size to 100% and display resolution to 1920 x 1080.
3. To run the program simply open the galton.py file and run the code on Visual Studio Code (or any other preferred IDE). Once the program is run, this window should appear and the program will start automatically.



4. On the right are several buttons. There are 3 options to change the properties of the simulation, the first of which is the ball quantities buttons which either increase or decrease the amount of balls, the second of which is the ball mass buttons which increase or decrease the mass of the balls, and lastly there's the ball radius buttons which increase or decrease the size of the balls either by pressing the + or - buttons respectively. The restart button at the top restarts the program while maintaining the altered variables of the balls while the reset button below it restarts the program and also resets all the variables of the simulation back to its default values.
5. **Additional information:**

- Ball quantities: +50 or -50 (it will increase the quantity of balls spawned by 50 for each time the user clicks on the + button and decrease by 50 for each time the user clicks on the - button.)
- Ball mass: +5 or -5 (it will increase the mass by 5 for each time the user clicks on the + button and decrease by 5 for each the user clicks on the - button.)

- Ball radius: +0.25 or -0.25 (it will increase the balls radius size by 0.25 for each time the user clicks on the + button and will decrease by 0.25 for each time the user clicks on the - button)

a) Github Link

https://github.com/Pandalmation/Galton-Simulation

b) PowerPoint Slides

Project Proposal:
https://www.canva.com/design/DAFmKcMYQBA/z0xxO3aT6JHsLZykomYk2g/view?utm_content=DAFmKcMYQBA&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink
Project Milestone:
https://www.canva.com/design/DAFmKQr_kJY/BxaL_kIfZ2TsuKaIkNRmvA/view?utm_content=DAFmKQr_kJY&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink
Final Project:
https://www.canva.com/design/DAFk78IzGVo/2kFCuKK9iMVCDbfl6vQ1hQ/view?utm_content=DAFk78IzGVo&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink

c) Video

https://youtu.be/UfVuEI82e6o

d) Code

The code is inside the github repository and is named galton.py