A

REPORT

ON

OOP Project

By

Sanyam Shah	**2019A3PS0287P**
Vasu Swaroop	**2019B4A70656P**

Topic

Spring Forces and Friction (Project no. 2)

Group no. - 103

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE,

PILANI

(Rajasthan)

Acknowledgements

We would like to express our deepest appreciation to all those whose efforts made this project possible. Special gratitude to our professors, Dr. Amit Dua and Dr. Amitesh Singh Rajput, whose constant guidance, suggestions and encouragement helped us to complete this project. Furthermore, we would also like to acknowledge with much appreciation the crucial role of the teaching assistant Aryaman, who helped clear our doubts and guided us throughout the making of the project.

Last but not the least, many thanks to our batchmates who cleared our doubts and supported us throughout the project.

Table of content

Description of Classes

We have created 4 different packages to handle different tasks in the project. Below is the description of each package and its member classes and interfaces along with the description of the Driver class.

1. Objt Package
   - ➢ Objects

     This is the superclass of all the physical objects, it contains all the common attributes like ObjectID, ObjectTypeID, mass, friction coefficient (mu), arrays for initial displacement, velocity and arrays for current displacement, velocity and acceleration declared as private variables. It has a constructor which takes all the attributes as a function parameter and creates an object initialized with those values, it also has the getter and setter methods for all the attributes. It has a toString method to print the output for any object of the class.

   - ➢ Rollable

     It is an abstract class which extends the Objects class, and is used as a superclass for all the rollable physical objects :- sphere and cylinder.

   - ➢ Non_Rollable

     It is an abstract class which extends the Objects class, and is used as a superclass for all the non-rollable physical objects :- solid block.

   - ➢ Sphere

     It is the concrete class for a sphere, it extends Rollable, it has a private variable- radius and the getter and setter methods for radius. It also has a constructor which calls the super() constructor and then initializes the radius variable.

   - ➢ Cylinder

     It is the concrete class for a cylinder, it extends Rollable, it has private variables- radius and height and the getter and setter methods for them. It also has a constructor which calls the super() constructor and then initializes the radius and height variables.

- ➢ Solid_Block

  It is the concrete class for a solid block, it extends Non_Rollable, it has private variables- length and width and the getter and setter methods for them. It also has a constructor which calls the super() constructor and then initializes the length and height variables.

- ➢ Environment

  It is an abstract class which is extended by all the environments like wedge and spring.

- ➢ Wedge

  It is the concrete class for a wedge, it extends Environment, has a private variable - angle and the getter, setter methods for it. It also has a function to describe the constraints for a wedge system.

- ➢ Spring

  It is the concrete class for a spring, it extends Environment, has a private variable - k (spring constant) and the getter, setter methods for it. It also has a function to describe the constraints for a spring.

2. Input Package
   - ➢ Input

     It is the abstract class for input and is used to get the common input from the user.

   - ➢ Output

     It is the class to display output for all systems.

   - ➢ Collison_Input

     It is the concrete class to get the specific input for the Collision system.

   - ➢ Wedge_Input

     It is the concrete class to get the specific input for the Wedge system.

- Pulley_Input

  It is the concrete class to get the specific input for the Pulley system.

- Spring_Attached_Input

  It is the concrete class to get the specific input for the Spring (object attached) system.

3. Sys Package

- Systems

  It is the superclass for all systems, and contains methods to update the velocity and the distance of the physical objects.

- Collision

  It is the concrete class for the collision system, it extends Systems and implements Force_X, Ground and Friction. It gets the input from Collision_Input. It has the simulate() method which simulates the collision system by using different functions from the superclass and interfaces.

- Wedge

  It is the concrete class for the Wedge system, it extends Systems and implements Force_X, Slope and Friction. It gets the input from Wedge_Input. It has the simulate() method which simulates the wedge system by using different functions from the superclass and interfaces.

- PulleySys

  It is the concrete class for the Pulley system, it extends Systems and implements Force_Y and Pulley. It gets the input from Pulley_Input. It has the simulate() method which simulates the Pulley system by using different functions from the superclass and interfaces.

- Spring_Attached

  It is the concrete class for the Spring with object attached system, it extends Systems and implements Spring_Force and Ground. It gets the input from Spring_Attached_Input. It has the simulate() method which simulates the Spring system by using different functions from the superclass and interfaces.

4. Describe Package

This package has the following interfaces which are implemented by the different system classes in the Sys package:-

- ➔ Display
- ➔ Force_X
- ➔ Force_Y
- ➔ Forces
- ➔ Friction
- ➔ Ground
- ➔ Pulley
- ➔ Slope
- ➔ Spring_Force

5. Driver Class

   This class has the main() method, wherein we first present a Menu of the systems with their descriptions to the user, and then create the object of the chosen system, take the input and provide it to the system and then print the output for the system.

Sample Test Cases

1. Collision

a. Features

i. The objects can’t be placed in an intersecting position. Basically, we have assumed that the objects have volume and can’t merge into each other. We have made a point of collision variable for that. We have also left a tolerance as a margin of error.

ii. It is an N-Body system, where we determine the properties for all the bodies (initial velocity, friction, and physical dimensions), and then model the interactions between them. N can also be 1.

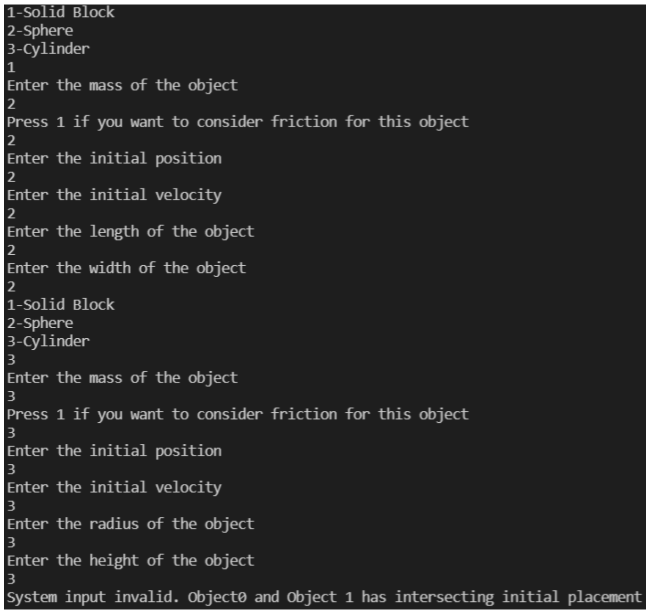
iii. We have assumed same mass of the objects, which results in switching of the velocity (the maths was simpler)

iv. The collision condition is similar to the first feature. At each iteration n body check for collision given a tolerance limit.

v. Friction can also be added. The condition for checking friction is to check if the object is moving or not.

b. Test Cases

i. Take radius as 2, initial placement as 2 for body 1. Radius 3, initial placement 3 for body 2.



```
1-Solid Block
2-Sphere
3-Cylinder
1
Enter the mass of the object
2
Press 1 if you want to consider friction for this object
2
Enter the initial position
2
Enter the initial velocity
2
Enter the length of the object
2
Enter the width of the object
2
1-Solid Block
2-Sphere
3-Cylinder
3
Enter the mass of the object
3
Press 1 if you want to consider friction for this object
3
Enter the initial position
3
Enter the initial velocity
3
Enter the radius of the object
3
Enter the height of the object
3
System input invalid. Object0 and Object 1 has intersecting initial placement
```

ii. Single Body System

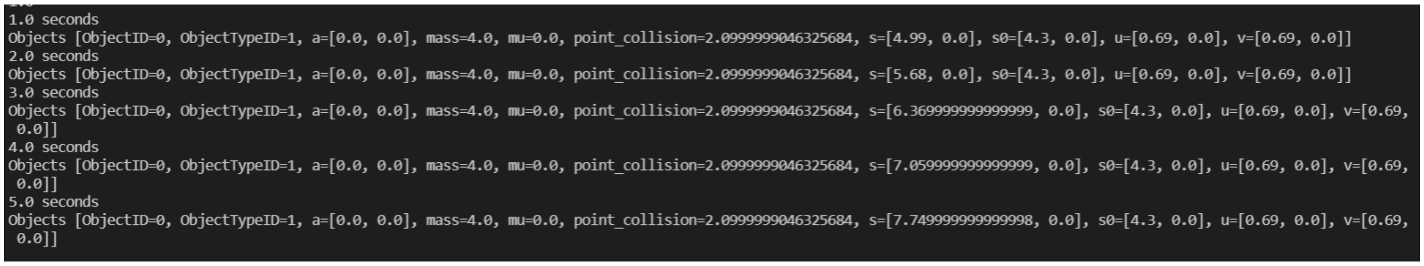
```
Number of objects you want to add?
1
1-Solid Block

2-Sphere

3-Cylinder

1
Enter the mass of the object (in Kg)
4
Press 1 if you want to consider friction for this object
2
Enter the initial position (in m)
4.3
Enter the initial velocity (in m/s)
0.69
Enter the length of the object (in m)
4.2
Enter the width of the object (in m)
```

2



```
1.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=4.0, mu=0.0, point_collision=2.0999999046325684, s=[4.99, 0.0], s0=[4.3, 0.0], u=[0.69, 0.0], v=[0.69, 0.0]]
2.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=4.0, mu=0.0, point_collision=2.0999999046325684, s=[5.68, 0.0], s0=[4.3, 0.0], u=[0.69, 0.0], v=[0.69, 0.0]]
3.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=4.0, mu=0.0, point_collision=2.0999999046325684, s=[6.369999999999999, 0.0], s0=[4.3, 0.0], u=[0.69, 0.0], v=[0.69,
 0.0]]
4.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=4.0, mu=0.0, point_collision=2.0999999046325684, s=[7.059999999999999, 0.0], s0=[4.3, 0.0], u=[0.69, 0.0], v=[0.69,
 0.0]]
5.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=4.0, mu=0.0, point_collision=2.0999999046325684, s=[7.749999999999998, 0.0], s0=[4.3, 0.0], u=[0.69, 0.0], v=[0.69,
 0.0]]
```

iii. n Body System

Number of objects you want to add?

4

1-Solid Block

2-Sphere

3-Cylinder

1

Enter the mass of the object (in Kg)

1

Press 1 if you want to consider friction for this object

2

Enter the initial position (in m)

3

Enter the initial velocity (in m/s)

10

Enter the length of the object (in m)

1

Enter the width of the object (in m)

1

1-Solid Block

2-Sphere

3-Cylinder

```
2
Enter the mass of the object (in Kg)
2
Press 1 if you want to consider friction for this object
2
Enter the initial position (in m)
6
Enter the initial velocity (in m/s)
-1
Enter the radius of the object (in m)
1
1-Solid Block

2-Sphere

3-Cylinder

3
Enter the mass of the object (in Kg)
6
Press 1 if you want to consider friction for this object
2
Enter the initial position (in m)
14
Enter the initial velocity (in m/s)
-1.5
Enter the radius of the object (in m)
1
Enter the height of the object (in m)
1
1-Solid Block
```

2-Sphere

3-Cylinder

3

Enter the mass of the object (in Kg)

2

Press 1 if you want to consider friction for this object

2

Enter the initial position (in m)

35

Enter the initial velocity (in m/s)

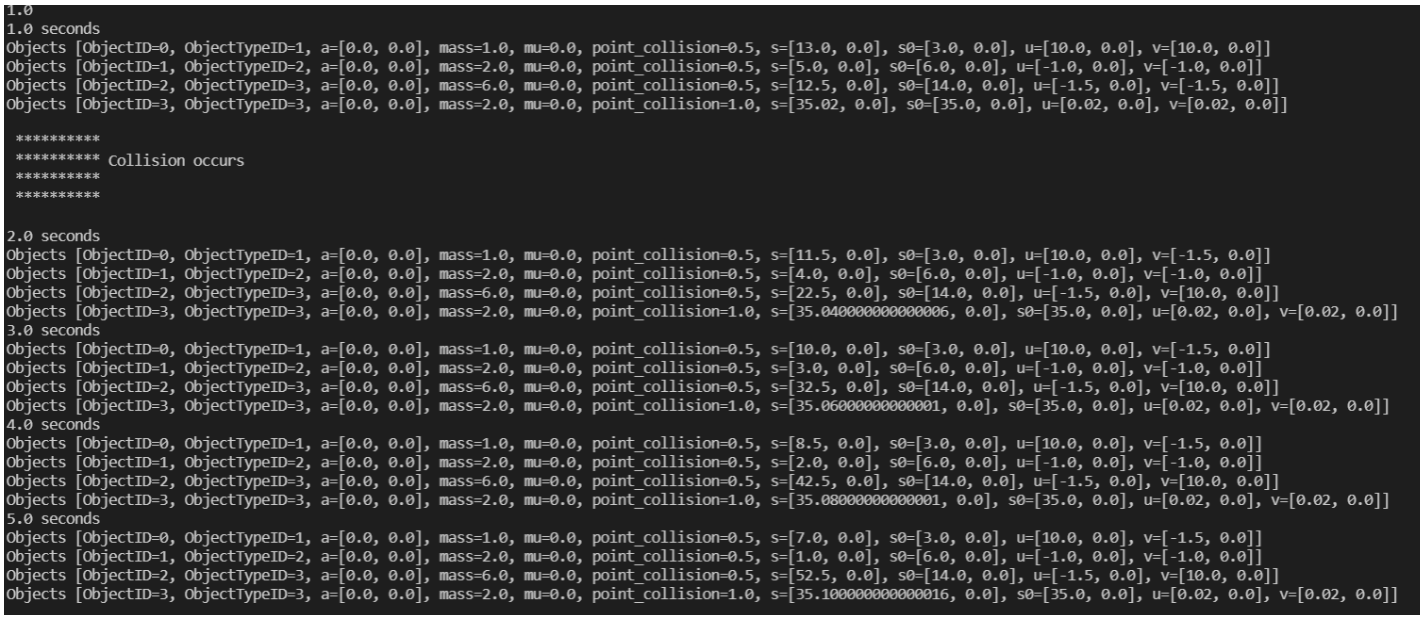
0.02

Enter the radius of the object (in m)

2

Enter the height of the object (in m)

2



```
1.0
1.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=1.0, mu=0.0, point_collision=0.5, s=[13.0, 0.0], s0=[3.0, 0.0], u=[10.0, 0.0], v=[10.0, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=0.5, s=[5.0, 0.0], s0=[6.0, 0.0], u=[-1.0, 0.0], v=[-1.0, 0.0]]
Objects [ObjectID=2, ObjectTypeID=3, a=[0.0, 0.0], mass=6.0, mu=0.0, point_collision=0.5, s=[12.5, 0.0], s0=[14.0, 0.0], u=[-1.5, 0.0], v=[-1.5, 0.0]]
Objects [ObjectID=3, ObjectTypeID=3, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[35.02, 0.0], s0=[35.0, 0.0], u=[0.02, 0.0], v=[0.02, 0.0]]

 **********
 ********** Collision occurs
 **********
 **********

2.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=1.0, mu=0.0, point_collision=0.5, s=[11.5, 0.0], s0=[3.0, 0.0], u=[10.0, 0.0], v=[-1.5, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=0.5, s=[4.0, 0.0], s0=[6.0, 0.0], u=[-1.0, 0.0], v=[-1.0, 0.0]]
Objects [ObjectID=2, ObjectTypeID=3, a=[0.0, 0.0], mass=6.0, mu=0.0, point_collision=0.5, s=[22.5, 0.0], s0=[14.0, 0.0], u=[-1.5, 0.0], v=[10.0, 0.0]]
Objects [ObjectID=3, ObjectTypeID=3, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[35.040000000000006, 0.0], s0=[35.0, 0.0], u=[0.02, 0.0], v=[0.02, 0.0]]
3.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=1.0, mu=0.0, point_collision=0.5, s=[10.0, 0.0], s0=[3.0, 0.0], u=[10.0, 0.0], v=[-1.5, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=0.5, s=[3.0, 0.0], s0=[6.0, 0.0], u=[-1.0, 0.0], v=[-1.0, 0.0]]
Objects [ObjectID=2, ObjectTypeID=3, a=[0.0, 0.0], mass=6.0, mu=0.0, point_collision=0.5, s=[32.5, 0.0], s0=[14.0, 0.0], u=[-1.5, 0.0], v=[10.0, 0.0]]
Objects [ObjectID=3, ObjectTypeID=3, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[35.06000000000001, 0.0], s0=[35.0, 0.0], u=[0.02, 0.0], v=[0.02, 0.0]]
4.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=1.0, mu=0.0, point_collision=0.5, s=[8.5, 0.0], s0=[3.0, 0.0], u=[10.0, 0.0], v=[-1.5, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=0.5, s=[2.0, 0.0], s0=[6.0, 0.0], u=[-1.0, 0.0], v=[-1.0, 0.0]]
Objects [ObjectID=2, ObjectTypeID=3, a=[0.0, 0.0], mass=6.0, mu=0.0, point_collision=0.5, s=[42.5, 0.0], s0=[14.0, 0.0], u=[-1.5, 0.0], v=[10.0, 0.0]]
Objects [ObjectID=3, ObjectTypeID=3, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[35.08000000000001, 0.0], s0=[35.0, 0.0], u=[0.02, 0.0], v=[0.02, 0.0]]
5.0 seconds
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, 0.0], mass=1.0, mu=0.0, point_collision=0.5, s=[7.0, 0.0], s0=[3.0, 0.0], u=[10.0, 0.0], v=[-1.5, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=0.5, s=[1.0, 0.0], s0=[6.0, 0.0], u=[-1.0, 0.0], v=[-1.0, 0.0]]
Objects [ObjectID=2, ObjectTypeID=3, a=[0.0, 0.0], mass=6.0, mu=0.0, point_collision=0.5, s=[52.5, 0.0], s0=[14.0, 0.0], u=[-1.5, 0.0], v=[10.0, 0.0]]
Objects [ObjectID=3, ObjectTypeID=3, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[35.100000000000016, 0.0], s0=[35.0, 0.0], u=[0.02, 0.0], v=[0.02, 0.0]]
```

2. Object on a Wedge

a. Features

i. Since friction is added, the dynamics of an object placed on a wedge is simulated.
ii. If the coefficient of friction is greater than tan(wedge_angle), the object doesn't move.
iii. In case the angle is lesser, friction will act and change the accelerations along the X and the Y axis.
iv. The condition for friction is, the object should be moving.
v. In case the net initial force is zero with zero initial velocity, the object remains at rest

b. Test Cases

i. Enter the mass of the object

   1

   Press 1 if you want to consider friction for this object

   2

   Enter the radius of the object

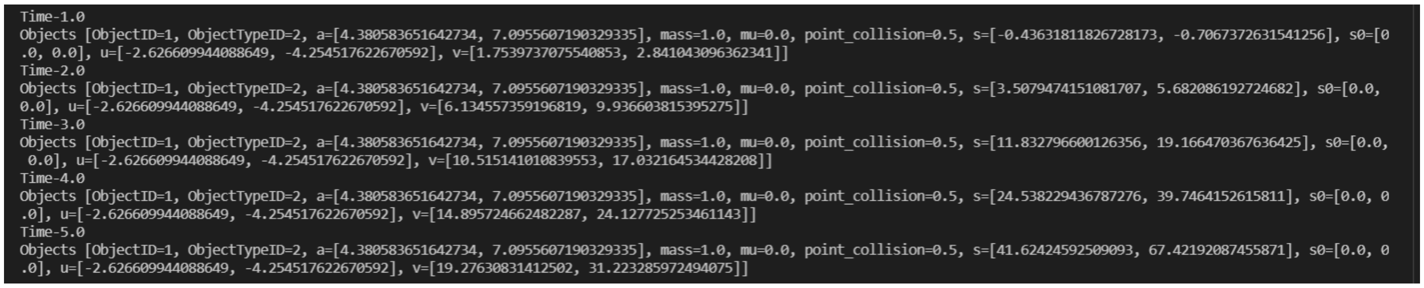
   1

   Enter the angle of inclination for the wedge

   45

   Enter the velocity.Positive velocity means down the slope. Negative means up the slope

   -5



```
Time-1.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.380583651642734, 7.0955607190329335], mass=1.0, mu=0.0, point_collision=0.5, s=[-0.43631811826728173, -0.7067372631541256], s0=[0
.0, 0.0], u=[-2.626609944088649, -4.254517622670592], v=[1.7539737075540853, 2.841043096362341]]
Time-2.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.380583651642734, 7.0955607190329335], mass=1.0, mu=0.0, point_collision=0.5, s=[3.5079474151081707, 5.682086192724682], s0=[0.0,
0.0], u=[-2.626609944088649, -4.254517622670592], v=[6.134557359196819, 9.936603815395275]]
Time-3.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.380583651642734, 7.0955607190329335], mass=1.0, mu=0.0, point_collision=0.5, s=[11.832796600126356, 19.166470367636425], s0=[0.0,
 0.0], u=[-2.626609944088649, -4.254517622670592], v=[10.515141010839553, 17.032164534428208]]
Time-4.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.380583651642734, 7.0955607190329335], mass=1.0, mu=0.0, point_collision=0.5, s=[24.538229436787276, 39.7464152615811], s0=[0.0, 0
.0], u=[-2.626609944088649, -4.254517622670592], v=[14.895724662482287, 24.127725253461143]]
Time-5.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.380583651642734, 7.0955607190329335], mass=1.0, mu=0.0, point_collision=0.5, s=[41.62424592509093, 67.42192087455871], s0=[0.0, 0
.0], u=[-2.626609944088649, -4.254517622670592], v=[19.27630831412502, 31.223285972494075]]
```

ii. Enter the mass of the object

   1

   Press 1 if you want to consider friction for this object

   1

Enter the coefficient of friction

0.01

Enter the radius of the object

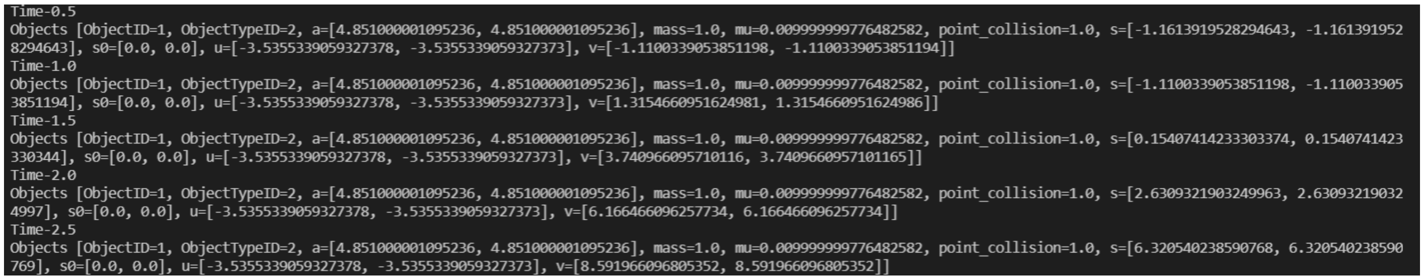
2

Enter the angle of inclination for the wedge

45

Enter the velocity.Positive velocity means down the slope. Negative means up the slope

-5



```
Time-0.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=1.0, s=[-1.161391952
8294643], s0=[0.0, 0.0], u=[-3.5355339059327378, -3.5355339059327373], v=[-1.1100339053851198, -1.1100339053851194]]
Time-1.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=1.0, s=[-1.1100339053851198, -1.110033905
3851194], s0=[0.0, 0.0], u=[-3.5355339059327378, -3.5355339059327373], v=[1.3154660951624981, 1.3154660951624986]]
Time-1.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=1.0, s=[0.1540741423303374, 0.1540741423
330344], s0=[0.0, 0.0], u=[-3.5355339059327378, -3.5355339059327373], v=[3.740966095710116, 3.7409660957101165]]
Time-2.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=1.0, s=[2.6309321903249963, 2.63093219032
4997], s0=[0.0, 0.0], u=[-3.5355339059327378, -3.5355339059327373], v=[6.166466096257734, 6.166466096257734]]
Time-2.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=1.0, s=[6.320540238590768, 6.320540238590
769], s0=[0.0, 0.0], u=[-3.5355339059327378, -3.5355339059327373], v=[8.591966096805352, 8.591966096805352]]
```

iii. Enter the mass of the object

1

Press 1 if you want to consider friction for this object

1

Enter the coefficient of friction

0.01

Enter the length of the object

1

Enter the width of the object

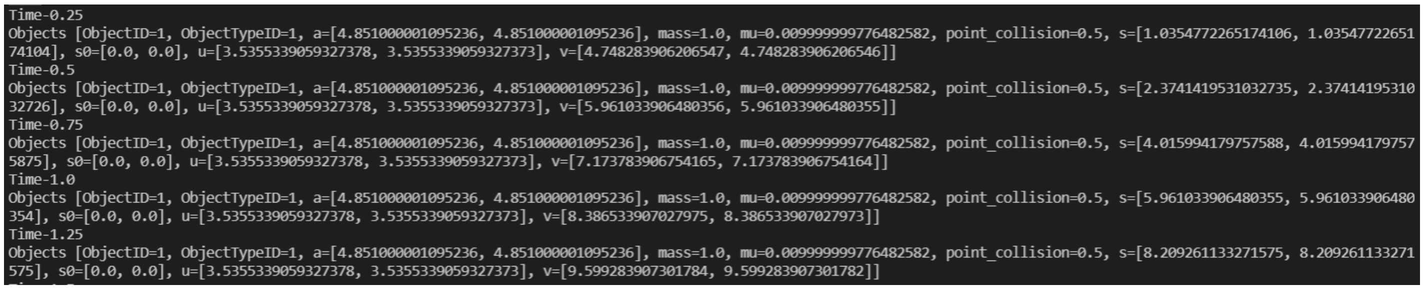
1

Enter the angle of inclination for the wedge

45

Enter the velocity.Positive velocity means down the slope. Negative means up the slope

5



```
Time-0.25
Objects [ObjectID=1, ObjectTypeID=1, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=0.5, s=[1.0354772265174106, 1.03547722651
74104], s0=[0.0, 0.0], u=[3.5355339059327378, 3.5355339059327373], v=[4.748283906206547, 4.748283906206546]]
Time-0.5
Objects [ObjectID=1, ObjectTypeID=1, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=0.5, s=[2.3741419531032735, 2.37414195310
32726], s0=[0.0, 0.0], u=[3.5355339059327378, 3.5355339059327373], v=[5.961033906480356, 5.961033906480355]]
Time-0.75
Objects [ObjectID=1, ObjectTypeID=1, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=0.5, s=[4.015994179757588, 4.015994179757
5875], s0=[0.0, 0.0], u=[3.5355339059327378, 3.5355339059327373], v=[7.173783906754165, 7.173783906754164]]
Time-1.0
Objects [ObjectID=1, ObjectTypeID=1, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=0.5, s=[5.961033906480355, 5.961033906480
354], s0=[0.0, 0.0], u=[3.5355339059327378, 3.5355339059327373], v=[8.386533907027975, 8.386533907027973]]
Time-1.25
Objects [ObjectID=1, ObjectTypeID=1, a=[4.851000001095236, 4.851000001095236], mass=1.0, mu=0.009999999776482582, point_collision=0.5, s=[8.209261133271575, 8.209261133271
575], s0=[0.0, 0.0], u=[3.5355339059327378, 3.5355339059327373], v=[9.599283907301784, 9.599283907301782]]
```

iv. Enter the mass of the object

    1

    Press 1 if you want to consider friction for this object

    1

    Enter the coefficient of friction

    0.9

    Enter the length of the object

    2

    Enter the width of the object

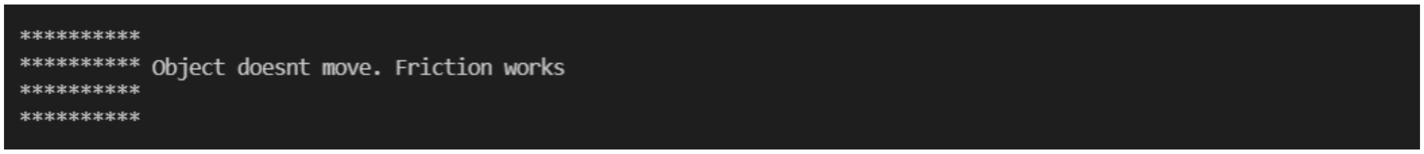
    2

    Enter the angle of inclination for the wedge

    30

    Enter the velocity.Positive velocity means down the slope. Negative means up the slope

    -5



```
**********
********** Object doesnt move. Friction works
**********
**********
```

3. Spring Attached System

a. Features

i. The object simulates simple harmonic motion for any compression/extension.

b. Test Cases

i. Enter the mass of the object (in kg)

   2

Enter the initial Displacement (in m)

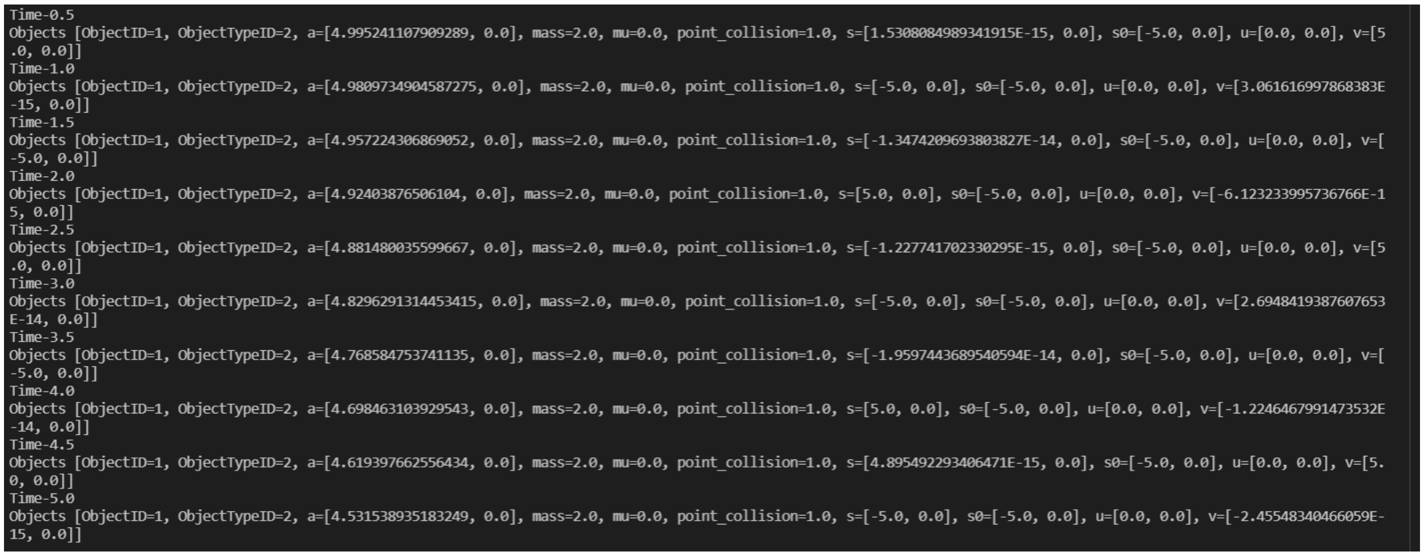
-5

Enter the Spring coefficient for the spring (in N/m)

50

Enter the radius of the object (in m)

2



```
Time-0.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.995241107909289, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[1.5308084989341915E-15, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[5
.0, 0.0]]
Time-1.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.9809734904587275, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-5.0, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[3.061616997868383E
-15, 0.0]]
Time-1.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.957224306869052, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-1.3474209693803827E-14, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[
-5.0, 0.0]]
Time-2.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.92403876506104, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[5.0, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[-6.123233995736766E-1
5, 0.0]]
Time-2.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.881480035599667, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-1.227741702330295E-15, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[5
.0, 0.0]]
Time-3.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.8296291314453415, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-5.0, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[2.6948419387607653
E-14, 0.0]]
Time-3.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.768584753741135, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-1.9597443689540594E-14, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[
-5.0, 0.0]]
Time-4.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.698463103929543, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[5.0, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[-1.2246467991473532E
-14, 0.0]]
Time-4.5
Objects [ObjectID=1, ObjectTypeID=2, a=[4.619397662556434, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[4.895492293406471E-15, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[5.
0, 0.0]]
Time-5.0
Objects [ObjectID=1, ObjectTypeID=2, a=[4.531538935183249, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-5.0, 0.0], s0=[-5.0, 0.0], u=[0.0, 0.0], v=[-2.45548340466059E-
15, 0.0]]
```

ii. 1-Solid Block

2-Sphere

3-Cylinder

3

Enter the mass of the object (in kg)

2

Enter the initial Displacement (in m)

2

Enter the Spring coefficient for the spring (in N/m)

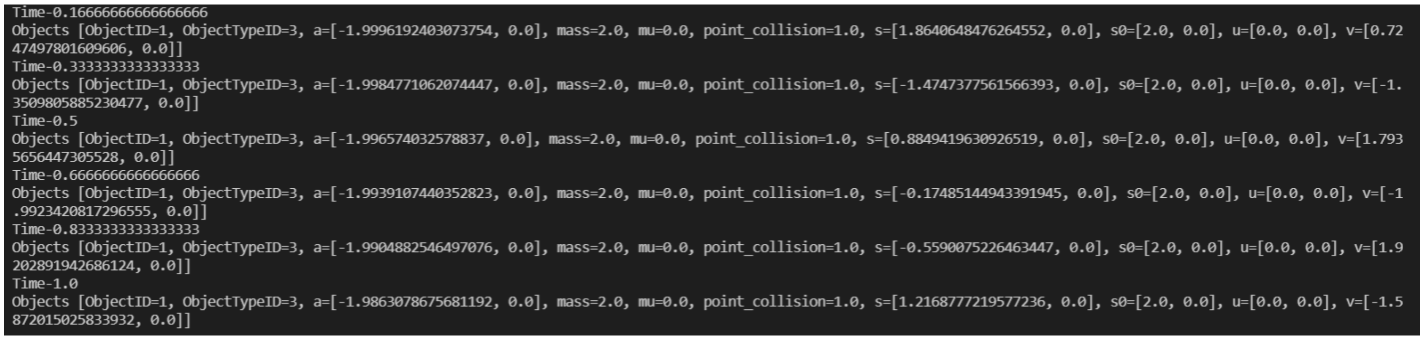
90

Enter the radius of the object (in m)

2

Enter the height of the object (in m)

2



```
Time-0.16666666666666666
Objects [ObjectID=1, ObjectTypeID=3, a=[-1.9996192403073754, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[1.8640648476264552, 0.0], s0=[2.0, 0.0], u=[0.0, 0.0], v=[0.72
47497801609606, 0.0]]
Time-0.3333333333333333
Objects [ObjectID=1, ObjectTypeID=3, a=[-1.9984771062074447, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-1.4747377561566393, 0.0], s0=[2.0, 0.0], u=[0.0, 0.0], v=[-1.
3509805885230477, 0.0]]
Time-0.5
Objects [ObjectID=1, ObjectTypeID=3, a=[-1.996574032578837, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.8849419630926519, 0.0], s0=[2.0, 0.0], u=[0.0, 0.0], v=[1.793
5656447305528, 0.0]]
Time-0.6666666666666666
Objects [ObjectID=1, ObjectTypeID=3, a=[-1.9939107440352823, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-0.17485144943391945, 0.0], s0=[2.0, 0.0], u=[0.0, 0.0], v=[-1
.9923420817296555, 0.0]]
Time-0.8333333333333333
Objects [ObjectID=1, ObjectTypeID=3, a=[-1.9904882546497076, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[-0.5590075226463447, 0.0], s0=[2.0, 0.0], u=[0.0, 0.0], v=[1.9
202891942686124, 0.0]]
Time-1.0
Objects [ObjectID=1, ObjectTypeID=3, a=[-1.9863078675681192, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[1.2168777219577236, 0.0], s0=[2.0, 0.0], u=[0.0, 0.0], v=[-1.5
872015025833932, 0.0]]
```

4. Pulley System

a. Features

i. Only a positive initial position can be provided

ii. We have added a clause, where the string ended and simulation ended. When one object reaches the pulley.

iii. We have imposed the restriction that string is of finite length and have calculated the string length

iv. The physics was simple. And we have verified it by showing that both the bodies have the same acceleration and velocity differing in direction.

v. Since it's a numerical solution of the Equations, we have added a tolerance value to all our calculations for checking simulation ending

b. Sample cases

i. Positive length for the String-

Enter the mass of the object

2

Enter the initial position

-3

Can only take positive values

Enter the initial position

-3

Can only take positive values

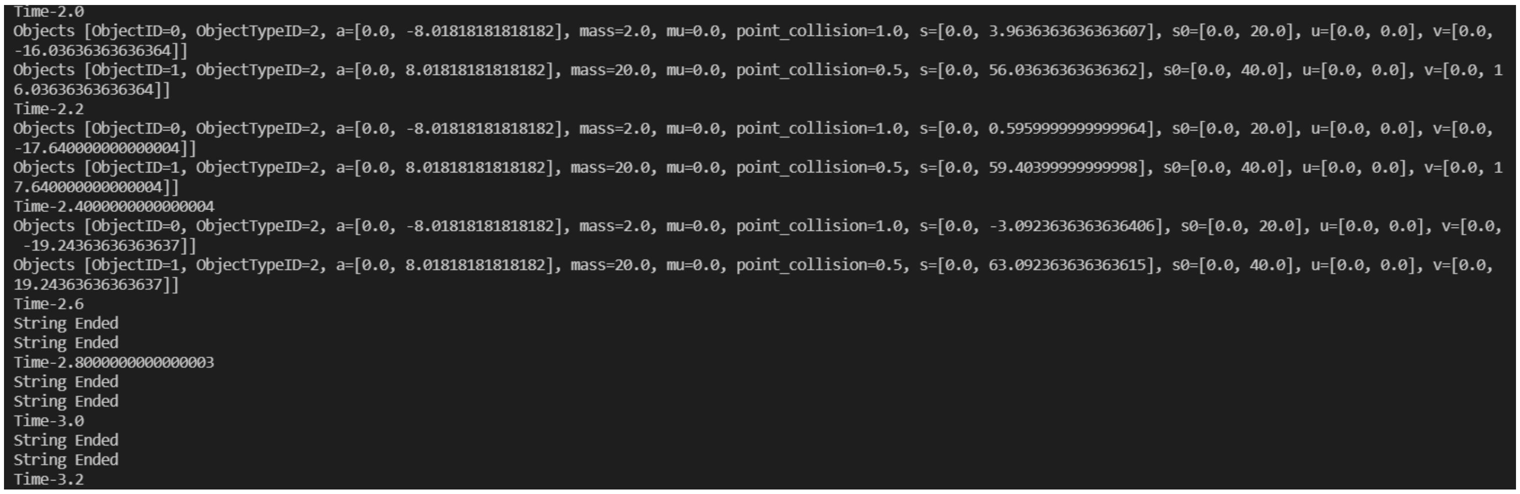
Enter the initial position

10

ii. String End Clause- String is finite

```
Input Time
20
Number of iterations
100
1-Solid Block
2-Sphere
3-Cylinder
2
Enter the mass of the object
2
Enter the initial position
20
Enter the radius of the object
2
1-Solid Block
2-Sphere
3-Cylinder
2
Enter the mass of the object
20
Enter the initial position
40
Enter the radius of the object
1
```

Sample Output



```
Time-2.0
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, -8.01818181818182], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 3.9636363636363607], s0=[0.0, 20.0], u=[0.0, 0.0], v=[0.0,
-16.03636363636364]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 8.01818181818182], mass=20.0, mu=0.0, point_collision=0.5, s=[0.0, 56.03636363636362], s0=[0.0, 40.0], u=[0.0, 0.0], v=[0.0, 1
6.03636363636364]]
Time-2.2
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, -8.01818181818182], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 0.5959999999999964], s0=[0.0, 20.0], u=[0.0, 0.0], v=[0.0,
-17.640000000000004]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 8.01818181818182], mass=20.0, mu=0.0, point_collision=0.5, s=[0.0, 59.40399999999998], s0=[0.0, 40.0], u=[0.0, 0.0], v=[0.0, 1
7.640000000000004]]
Time-2.4000000000000004
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, -8.01818181818182], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, -3.092363636363636406], s0=[0.0, 20.0], u=[0.0, 0.0], v=[0.0,
 -19.243636363636637]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 8.01818181818182], mass=20.0, mu=0.0, point_collision=0.5, s=[0.0, 63.092363636363615], s0=[0.0, 40.0], u=[0.0, 0.0], v=[0.0,
19.243636363636637]]
Time-2.6
String Ended
String Ended
Time-2.8000000000000003
String Ended
String Ended
Time-3.0
String Ended
String Ended
Time-3.2
```

iii. Standard case

1-Solid Block

2-Sphere

3-Cylinder

1

Enter the mass of the object

2

Enter the initial position

10

Enter the length of the object

3

Enter the width of the object

3

1-Solid Block

2-Sphere

3-Cylinder

2

Enter the mass of the object

5

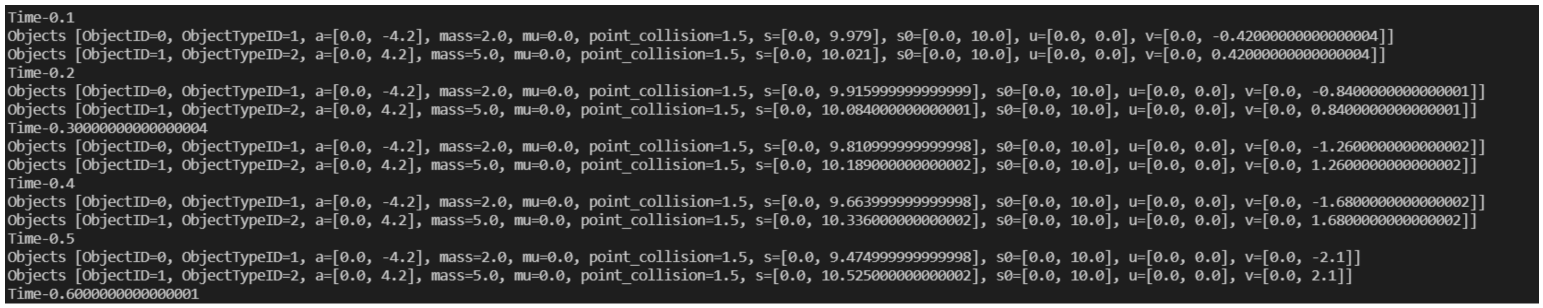
Enter the initial position

10

Enter the radius of the object

3

Sample Output



```
Time-0.1
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, -4.2], mass=2.0, mu=0.0, point_collision=1.5, s=[0.0, 9.979], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, -0.42000000000000004]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 4.2], mass=5.0, mu=0.0, point_collision=1.5, s=[0.0, 10.021], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 0.42000000000000004]]
Time-0.2
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, -4.2], mass=2.0, mu=0.0, point_collision=1.5, s=[0.0, 9.915999999999999], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, -0.8400000000000001]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 4.2], mass=5.0, mu=0.0, point_collision=1.5, s=[0.0, 10.084000000000001], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 0.8400000000000001]]
Time-0.30000000000000004
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, -4.2], mass=2.0, mu=0.0, point_collision=1.5, s=[0.0, 9.810999999999998], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, -1.2600000000000002]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 4.2], mass=5.0, mu=0.0, point_collision=1.5, s=[0.0, 10.189000000000002], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 1.2600000000000002]]
Time-0.4
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, -4.2], mass=2.0, mu=0.0, point_collision=1.5, s=[0.0, 9.663999999999998], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, -1.6800000000000002]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 4.2], mass=5.0, mu=0.0, point_collision=1.5, s=[0.0, 10.336000000000002], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 1.6800000000000002]]
Time-0.5
Objects [ObjectID=0, ObjectTypeID=1, a=[0.0, -4.2], mass=2.0, mu=0.0, point_collision=1.5, s=[0.0, 9.474999999999998], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, -2.1]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 4.2], mass=5.0, mu=0.0, point_collision=1.5, s=[0.0, 10.525000000000002], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 2.1]]
Time-0.6000000000000001
```

iv. No change in movement as mass same

1-Solid Block

2-Sphere

3-Cylinder

2

Enter the mass of the object

2

Enter the initial position

2

Enter the radius of the object

2

1-Solid Block

2-Sphere

3-Cylinder

2

Enter the mass of the object

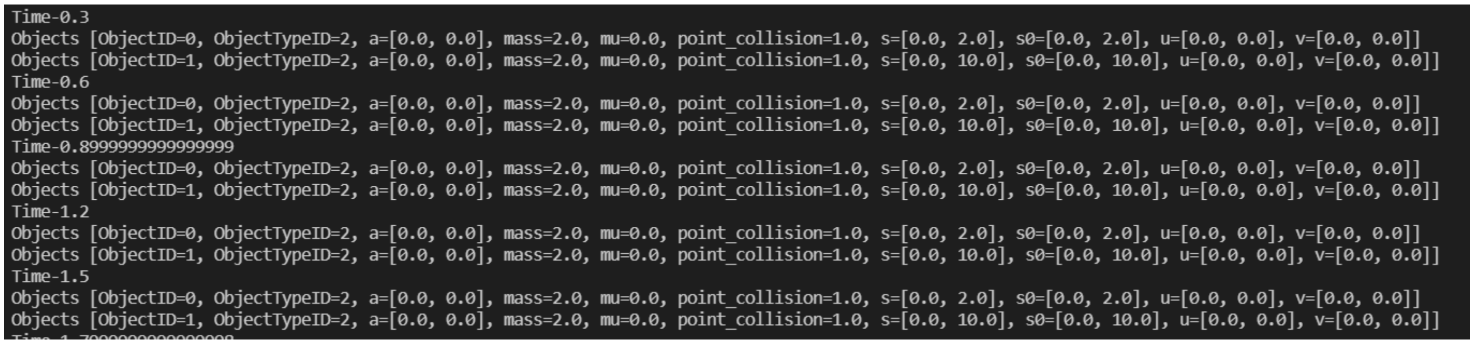
2

Enter the initial position

10

Enter the radius of the object

2



```
Time-0.3
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 2.0], s0=[0.0, 2.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 10.0], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Time-0.6
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 2.0], s0=[0.0, 2.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 10.0], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Time-0.8999999999999999
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 2.0], s0=[0.0, 2.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 10.0], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Time-1.2
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 2.0], s0=[0.0, 2.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 10.0], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Time-1.5
Objects [ObjectID=0, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 2.0], s0=[0.0, 2.0], u=[0.0, 0.0], v=[0.0, 0.0]]
Objects [ObjectID=1, ObjectTypeID=2, a=[0.0, 0.0], mass=2.0, mu=0.0, point_collision=1.0, s=[0.0, 10.0], s0=[0.0, 10.0], u=[0.0, 0.0], v=[0.0, 0.0]]
```

Known limitations/Bugs

- We have tried to be as representative of actual physical systems as possible. But we had taken some assumptions to ease the maths behind the scenes. Including them is possible but will take a little more time to implement. Some assumptions include
    - Same mass bodies in collision (We are inputting different masses, but we are assuming that collision happens with same mass assumption)
    - During Collision Coefficient of restitution is 1
    - We also assumed that a spring can be stretched/compressed arbitrarily
- We have imposed several restrictions on the input and output based on the system capabilities. Since we modelled using specific equations which changed based on the system, and since we didn't have the technical expertise of simulating vectors, we imposed these restrictions.
- We have created our system in a dynamic enough manner to include rolling, but couldn't implement it ourselves due to lack of time and complicated physics.
- We had some moderate success with GUI, but then decided to not use it within the scope of the project. Using it for simulating an n-body system would've complicated things a lot.
- Due to a little lack of time, we have not been able to optimise the input and output flow. The restrictions have sometimes been assumed implicitly.
- We have not been able to optimise the method scopes in the given project.
- Some methods have empty bodies to allow for inclusion of unimplemented methods of an interface.
- Although we have implemented the code for taking friction into account, due to faulty implementation of physics, we are getting wrong output (in case of collision), but the code is working completely fine and the output can be corrected by correcting the physics for friction, which became too complex for us to solve. In due time, we will be able to account for the error in our calculations and correct for the issues.

Future Work Possible

Our project is on physics simulation which in itself is a very vast topic and the interaction between different objects and environments has limitless possibilities. Although we have created a very basic model with some ideal assumptions due to our lack of expertise in the physics and mathematics required for the simulation, we have applied OOP concepts meticulously so that this project can be easily expanded and be taken to any level of complexity without making any changes to the already existing structure.

After adding rolling motion in this project, it can be modelled to make for an excellent physics engine which could serve the purpose of solving physics problems depending on the user's need for objects and environments. But even here some constraints on the input will be imposed by the programmer to ensure the integrity of the program is maintained.

Another feature that can be added is the GUI for the simulation, we could not add this due to our lack of knowledge of GUI programming, but with GUI implemented the project could be used as an educational tool for physics students, where they can see the interaction between different systems and environments in an animated format.

Analysis of the Project (OOP Principles)

Our project is a physics simulation which has 4 systems :-

1. N- body collision
2. Body on a wedge
3. Pulley System
4. Spring with object attached

We have extensively applied the concepts of OOP in our project, the following section contains detailed analysis of our project with respect to OOP concepts, identifying and justifying the concepts that we have applied or have not applied.

Major OOP principles which we are analysing in this section:-

1. **Encapsulate what varies**
2. **Favour composition over inheritance**
3. **Program to interface, not implementations**
4. **Strive for loosely coupled designs between objects that interact with each other**
5. **Classes should be open for extension but closed for modification**

Encapsulating the variable attributes

In our project, the variable attributes are limited, we could only find one such distinction which is whether an object is rollable or non rollable, we have made abstract classes for both of these behaviors and the objects can inherit from either of these classes.

Favour Composition over Inheritance

The different systems in our project are using different physical objects, so instead of creating different concrete classes with different objects as their superclass we have used composition, we are using a List with the reference to the Objects class which will instantiate the required type of object at the runtime therefore reducing the length of code and avoiding use of redundant code.

Program to an Interface

We have extensively used this concept of OOP in our project.

- We have created many different physical objects such as Sphere, Cylinder, Solid block etc, and all our simulation systems can take any physical object as the input, so here we are assigning the concrete implementation at runtime by creating an List with the reference to Objects which is the superclass of all the different objects and instantiating the required object type at runtime using a switch case.
- We have created many different interfaces such as ground, friction, pulley, force_x , force_y, slope, spring etc. such that any new system which needs any of these objects can easily be created by implementing the required interfaces. This concept helps us make our project easily extensible by any other programmer that wishes to add new features and thus reflects the beauty of Object Oriented Programming.
- We have applied the same principle while creating different physical objects such as sphere, solid block, cylinder etc. Here these objects can either be rollable or non-rollable so we have made abstract classes named Rollable and NonRollable which will contain appropriate attributes and methods, In our project we have left these classes to be empty because of the increased complexity in solving the physics for rolling motion, but these abstract classes indicate that any other programmer (with strong physics) can easily add rolling motion to our simulation without changing the structure of the project.

Loosely Coupled design between objects that interact with each other

All of our different systems use different physical objects such as sphere, block, spring, wedge etc. and all of the physical objects can be used in different systems (we have imposed some restrictions to reduce the complexity of physics equations but the possibility of using any object in any system exists).The advantages of our loosely coupled design are:-

- Changes to either a System or an Object will not affect the other, because the two are loosely coupled, we are free to make changes to either.
- Since our system depends only on the list objects, we can add new objects anytime, in fact we can replace any object with any other object at runtime.
- Likewise, we can remove objects anytime. We never need to modify the system to add new objects.
- We can reuse systems or objects independently from each other. If we have another use for a system or an object we can easily reuse them because the two aren’t tightly coupled.

Open for extension Closed for modification

We have superclasses like Objects and Systems which are open for extension but closed for modification, meaning that for any new object or system that we want to create we can extend these classes and make new concrete classes for the new object/system without changing the code for Objects or Systems class.

Analysis of the Project (Design Principles)

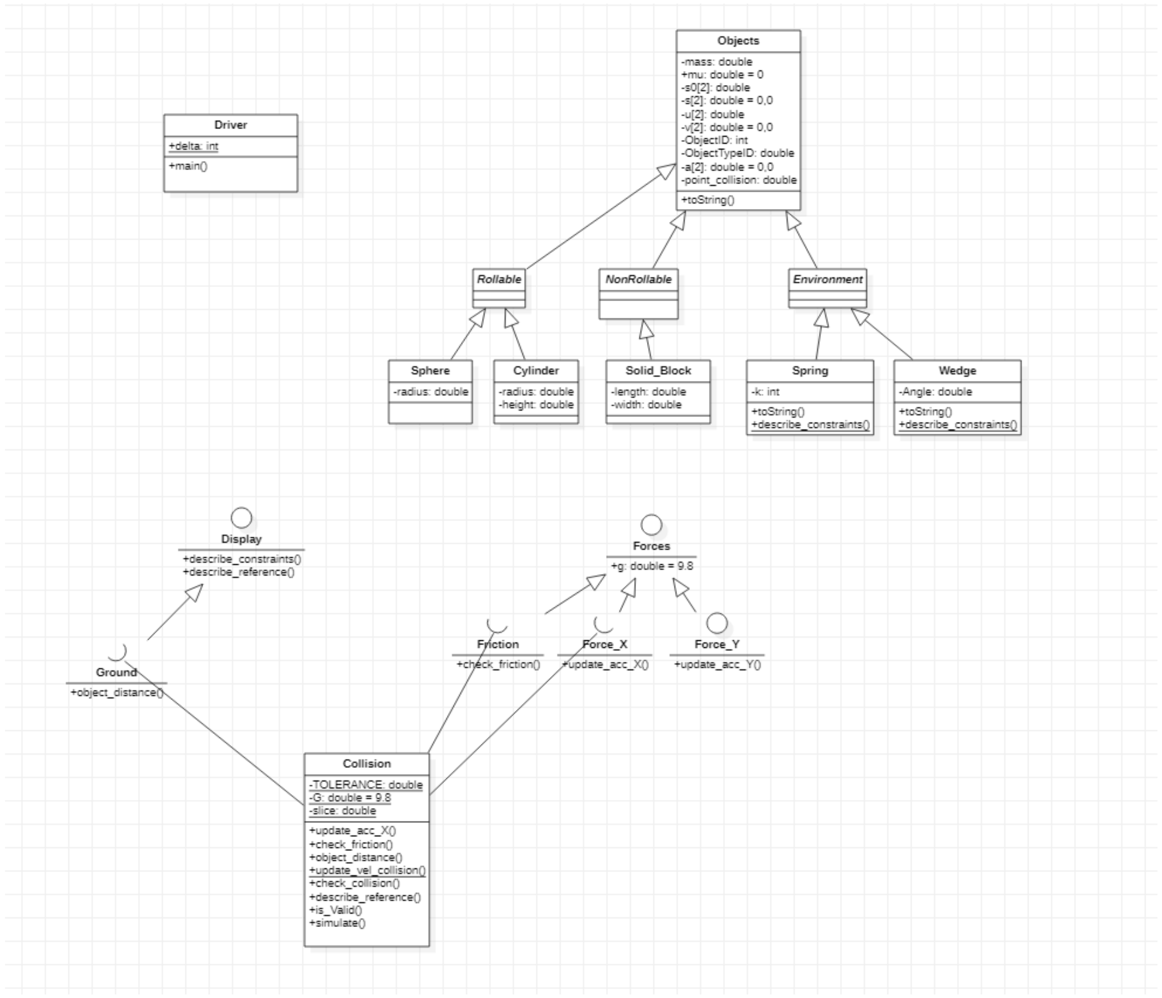
To our understanding our project doesn;t follow a single definite design pattern entirely. We have used some features from different patterns. For example, We have favoured composition over inheritance which is a distinct feature of Strategy pattern. Upon studying our architecture further, we came to the conclusion that our project most similarly represented by Factory Design Pattern (in a constricted extent)

Similarity with Factory Pattern

- We have also encapsulated object creation.
- We also have a parallel class hierarchy, for the Object classes and the System classes.
- We are letting the System decide which object it wants to create at runtime, therefore we are dependent upon abstraction and not on concrete classes.
- We haven't made the Objects class abstract because we were not taught design patterns during the time of making the project, but if we did then none of our variables would hold a reference to a concrete class.

UML Diagram

Since our project has many classes and interfaces, making a complete UML diagram is a very tedious task and also the diagram will be very complex to comprehend, so in this section we are showing the UML diagrams for a few classes which will give an overview of the working of our project.
We have also avoided adding the constructor, getter and setter functions in the UML class diagram to make it



shorter.
Here we have shown the class for Collision system, an object of this class will be created in the main() method in the driver class if the user selects to simulate the collision system from the menu shown in the main() method.
Please use the following link to get the .mdj file in case the UML diagram is not clearly visible in the report.
https://drive.google.com/drive/folders/10QIWyF_U4oUIyt11P6UrsKMJ4X4Vzc5f?usp=sharing

The code can also be accessed using the following github repo- https://github.com/Vhaatever/OOP-Project.git