**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY**



**PROJECT REPORT**

INTERACTIVE SIMULATION OF COMPOSITION OF FORCE

Course: Object Oriented Programming – Semester 2023.2

Supervisor: Prof. Tran The Hung

**Group 18**

|  |  |
| --- | --- |
| **Full name** | **Student ID** |
| Ngo Minh Trung | 20226004 |
| Phan Hoang Tu | 20226068 |
| Dang Trong Van | 20226072 |

1. **Assignment of members**

|  |  |  |
| --- | --- | --- |
| **Full name** | **Student ID** | **Contribution** |
| Ngo Minh Trung | 20226004 | GUI, class PhysicalObject, class Cylinder, class Force, class Cube, class Surface, Use case diagram, Report + Slide preparation |
| Phan Hoang Tu | 20226068 | Testing application on Terminal, General Class Diagram |
| Dang Trong Van | 20226072 | Detailed Class Diagram |

1. **Mini Project Description**

A simple interactive simulation for demonstrating Newton’s laws of motion.

1. **Requirements**

In the simulation, the user controls a physical system. The system includes three components: one main object, the surface (which is always horizontal) and an actor who can apply a horizontal force on the object:

* The main object: The user has two options for the main object, either a cube-shaped object or a cylinder-shaped object.
* The actor: The actor always applies force on the center of mass of the main object.
* The surface: The user can control the friction coefficients of the surface in the bottom right panel. There are two friction coefficients: static friction coefficient and kinetic friction coefficient. Note that the value of the static coefficients must be higher than the value of the kinetic coefficient.

We recalculate the position of the main object after each time interval Δt.

1. **Use Case Diagram**

A diagram of a diagram

Description automatically generated

The user can control all the components of the physical system and observe the motion of the main object, specifically:

* To set up the main object in the system, the user can drag an option from the object menu on the bottom left onto the surface, then click on the object, and provide the parameters in the context menu that pops up.
* To control the strength and direction of the applied force, the user can use the bottom center panel by using the sliding bar or specify the number of Newtons in the textbox.
* For each coefficient, the user can control its value through a sliding bar and a text box.

Throughout the motion simulation process, the user can

* Change the applied force as well as the friction coefficients of the surface.
* Pause, continue and reset the simulation.
* Choose to show or hide detailed information such as the forces, the sum of forces, the values of forces, the mass, speed and acceleration of the main object through the corresponding tick-boxes on the panel on the upper right:
* The forces and sum of forces are displayed as arrows.
* The masses are displayed as text on the main object.
* The speed and acceleration are displayed as text on the upper left corner.

1. **Design**
2. **General class diagram**

**A diagram of a computer program

Description automatically generated**

1. **Detailed class diagram**

**GUI class diagram**A diagram of a computer program

Description automatically generated

**Model package class diagram**

**A screenshot of a computer program

Description automatically generated**

1. **Explanation of class diagrams**

Class View is inherited from Application and overrides the start method by creating root node of scene graph which used to load the View.fxml file. To implement the user interface, we created Controller that implements the Initializable interface of fxml package. Controller overrides the initialize method, to set up the user interface and the initial state of the simulation whenever Controller class is initialized.

In the model package, we created an abstract class named PhysicalObject that contains the basic attributes and methods of an object in this simulation.

Since the users have two types of objects: a cube or a cylinder, we used the Inheritance technique here by creating two classes called Cube and Cylinder. These two classes are inherited from the PhysicalObject class. Polymorphism technique has also been used here as Cube and Cylinder object have different way to calculate Friction. Moreover, we need to consider angular motion of the Cylinder object while Cube objects don’t have such characteristics. Consequently, we need to override the method FrictionUpdate in these two classes and methods related to angular motion like: getAngularAcceleration, getAngularVelocity and get AngularPosition in class Cylinder.

The classes and their relationships can be described as follows:

* **PhysicalObject**: This is an abstract class that represents a physical object in the simulation. It contains properties such as mass, height, position, velocity, acceleration, and forces acting on it (applied force, friction, and normal force). It also contains methods to update these properties and calculate resultant force. This class is extended by **Cube** and **Cylinder,** which are specific types of physical objects.
* **Cube** and **Cylinder:** These classes extend PhysicalObject. They represent specific types of physical objects in the simulation. They inherit all properties and methods from **PhysicalObject** and can also have their own unique properties and methods.
* **Force:** This class represents a force in the simulation. It has a value property and can be used to set the value of different forces acting on the physical objects.
* **Surface:** This class represents the surface on which the physical objects are placed. It has properties for static and kinetic friction coefficients.
* **Controller:** This class implements the Initializable interface and controls the user interface and the simulation. It contains instances of **PhysicalObject, Surface,** and **Force** to represent the current state of the simulation. It also contains methods to handle user interactions and update the simulation.
* **Association/Aggregation/Composition** example**:**
* Cube and Cylinder are associated with PhysicalObject through inheritance. They are subclasses of PhysicalObject.
* Cube and Cylinder are associated with Surface and Force through composition. They contain instances of Surface and Force as their properties.
* Controller is associated with Cube, Cylinder, Surface, and Force because it uses these objects to control the simulation.

Important methods in these classes include:

* **PhysicalObject.update(double t):** This method updates the state of the physical object in each time step. It calls other methods to update friction, acceleration, velocity, and position of the object.
* **Cube.frictionUpdate()** and **Cylinder.frictionUpdate():** These methods override the f**rictionUpdate()** method in **PhysicalObject**. They calculate the friction force acting on the cube or cylinder based on the applied force, normal force, and the friction coefficients of the surface.
* **Controller.initialize(URL arg0, ResourceBundle arg1):** This method is called when the Controller is initialized. It sets up the user interface and the initial state of the simulation.
* **Controller.updateObject():** This method is called in each time step of the simulation. It updates the state of the physical object and the user interface.