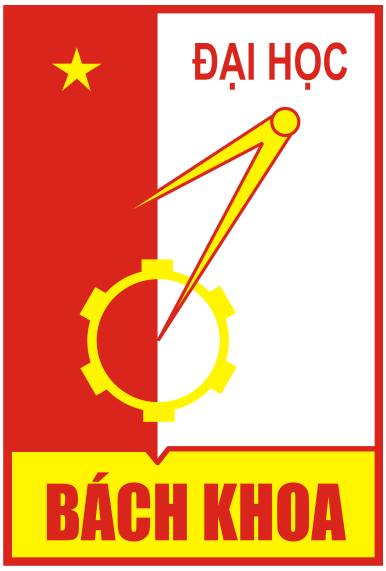
HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE

o0o



PROJECT REPORT:
Interactive simulation of the composition of forces

Course: Object-oriented Programming - IT3100E
Class Code: 141177
Supervisor: Associate Prof Nguyen Thi Thu Trang
Member of our group:

Vo Dinh Dat	20214890	@sis.hust.edu.vn
Tran Quoc De	20210179	@sis.hust.edu.vn
Nguyen Minh Cuong	20210140	@sis.hust.edu.vn
Hoang Thanh Dat	20214899	dat.ht214889@sis.hust.edu.vn

Ha Noi 2023

Abstract

This report presents the development of a simulation application for modeling object motion on a surface following Newton's rule of motion. The application is built using Java and JavaFX, following the Model-View-Controller (MVC) architectural pattern. Users can interact with the simulation by creating objects, applying forces, and observing the resulting motion. The application incorporates animations and provides real-time object statistics to enhance the user experience. The report highlights the key components and implementation details of the application, emphasizing its potential for educational purposes and interactive demonstrations. Overall, the simulation application offers a versatile tool for exploring motion and dynamics concepts.

Contents

1 Assignment of members

1.1 List of task

1. Vo Dinh Dat
    - Design 25%
    - model/object/MainObject 100
    - model/object/Cylinder 100%
    - model/force/ForceSimulation 25%
    - controller/AnimationController 70%
    - controller/DragDropController 50%
    - controller/MainSimulationController 70%
    - controller/WelcomePage.java 100%
    - view/MainSimulation.fxml 100%
    - view/Welcome.fxml 100%
    - view/ForcePanel.fxml 50%
2. Tran Quoc De
    - Design 25%
    - model/object/Cube 100%
    - model/surface 100%
    - controller/AnimationController 30%
    - controller/MainSimulationController 30%
    - controller/StatisticController 50%
    - controller/CheckboxController 100%
    - controller/SurfaceController 100%
    - view/StatsPanel 100%
    - view/Checkbox 100%
3. Nguyen Minh Cuong
    - Design 25%
    - model/force/AppliedFoce 100%
    - model/force/Force 100%
    - model/force/ForceSimulation 75%
    - model/force/FrictionForce 100%
    - model/force/Vector 100%
    - controller/forceController 100%
    - view/ForcePanel.fxml 50%

- view/MainSimulation.fxml 100

4. Hoang Thanh Dat
   - Design 25%
   - controller/DragDropController 50%
   - controller/StatisticController 50%
   - presentation 100
   - demo 100
   - report 100

2 Mini-project description

2.1 Overview

Newton's laws of motion form the foundation of classical mechanics, describing the behavior of objects in response to forces. The laws encompass three principles: the first law states that objects remain at rest or in uniform motion unless acted upon by an external force, the second law relates the force applied to an object with its mass and acceleration, and the third law states that every action has an equal and opposite reaction.

Our group aims to develop an interactive simulation application to showcase Newton's laws of motion. To facilitate collaboration and sharing, we will utilize Github for version control. The development process will involve creating use-case and class diagrams, incorporating object-oriented programming concepts like Inheritance, Polymorphism, and Association, which will be explained in our report and presentation along with the reasoning behind our chosen ideas for the project.

2.2 Mini-project requirement

As we mentioned before, in this project, our group needs to build a simple interactive simulation application for demonstrating Newton's laws of motion with some specific requirements:

- The program's graphical user interface (GUI) will resemble the reference [1], consisting of three primary components: the sky, the surface, and the main object. The user can have control over all the components of the physical system, primarily focusing on the main object and the surface. Additionally, the user can adjust a horizontal force on the object's center of mass. The user can then observe and analyze the resulting motion of the main object.
- On the main menu,
  - To enter the program, the user needs to click on the **Start** button.
- To initiate the application, the user will begin by dragging an option (a cube or a cylinder) from the object menu located at the bottom left of the interface. The user can then drag either a cube or a cylinder onto the surface. Subsequently, the user will be able to specify appropriate parameters corresponding to the chosen object's shape:
  - Cube object: side-length, mass.
  - Cylinder object: radius, mass.

- While the simulation is running, the user can control:
  - The applied horizontal force an be controlled by adjusting the force's magnitude and direction. This can be done through a slide bar or by entering a specific number in a text box located in the bottom center panel..
  - The user can modify the static and kinetic friction coefficients of the surface. This can be done by adjusting the value using a slide bar or by entering a specific value in a text box located in the bottom right panel.
  - All relevant figures, such as forces' values and directions, the net force, mass, velocity, acceleration, and position of the main object (including angular units for cylinder objects), can be monitored. The user can enable or disable the display of these statistics by selecting the corresponding tick-boxes located in the upper right panel.
  - User can change simulation state such as start/ continue, pause, and reset through "Play" or "Pause" buttons and "Reset" button
- In order to simulate motion, the program will recalculate the statistics of the main object after each time interval Δt and determine how the physical forces effect the object.
- To provide further clarification on the project:
  - The decision to restrict the user from choosing a different main object during the simulation was made based on the logical coherence of the simulation. Switching between different object types, such as from a cube to a cylinder or vice versa, would introduce inconsistencies in the statistics and units associated with the objects.
  - The user does not have the capability to modify any parameters of the current object. However, if time permits, this functionality may be added in future updates to enhance the flexibility of the simulation.
  - To overcome the limitations mentioned above, the user can reset the application, allowing them to choose the main object again and initiate a fresh simulation with the desired parameters. This reset feature ensures that the simulation remains coherent and provides an opportunity for the user to make any necessary changes to the main object if needed.

2.3 Use-case diagram explanation

2.3.1 Setting up Main Object

- During the setup of the main object, the user will have the option to choose its type by dragging and dropping a cube or a cylinder. After selecting the desired type, the user will need to provide values for the object's mass and side-length (or radius). These values will determine the physical characteristics of the main object within the simulation.
- Once the user has passed in all the parameters for the object, our program will create and modify the size of that object based on the provided values. The modified object will be displayed above the surface in the GUI. Additionally, the user will be able to observe how the object's size adjusts dynamically to accommodate changes in the window size, ensuring a responsive and visually appealing display.

Figure 1: Use-case diagram

2.3.2 Modifying Forces

- The user will have two options to adjust values: using a slider or directly typing into a text box. If the user chooses to type a value into the text box, they must press "Enter" to confirm the input.
- After the user modifies the applied force, our program will update the object with the new force and recalculate the net force, acceleration, and the width of the vector representing the force
- As a result of the updated applied force, the user will observe changes in the object's speed, reflected by its movement becoming faster or slower. Additionally, the length of the force vector will be adjusted accordingly. The labels representing acceleration and force will also be updated to reflect the current applied force value.
- Similarly, the user can adjust the friction force by using a slider or typing a value into a text box. If the user opts to type a value, they will need to press "Enter" to confirm the input.

2.3.3 Controlling Simulation State

- The user can modify the simulation state using buttons such as "Play" or "Pause" and the "Reset" button.
  - Pause: This button allows the user to temporarily pause the simulation, freezing the motion of the objects and temporarily halting any calculations or updates.
  - Continue: By clicking the "Play" button, the user can resume the simulation from the paused state, allowing the objects to continue their motion and updating the relevant calculations.
  - Reset: The Reset button enables the user to reset the simulation to its initial state. This action will clear any changes made to the objects, forces, or settings, allowing the user to start again with the default parameters and configurations.

2.3.4 Displaying Detailed Information

- The user will have the option to choose which statistic information to display by ticking checkboxes.
  - Forces: This checkbox allows the user to display the individual forces acting on the main object as colored arrows, indicating their directions. By selecting this checkbox, the user will be able to visualize the forces exerted on the main object and observe their respective directions. This feature enhances the understanding of the forces at play within the simulation.
  - Sum of forces: By selecting this checkbox, the user can view the total or net force acting on the main object. The net force represents the combined effect of all the forces acting on the object. Enabling this checkbox will display the direction of the net force, providing a clear indication of the overall force influencing the motion of the main object. This allows the user to assess the resultant force and understand its impact on the object's behavior.
  - Values: Enabling this checkbox will display the numerical values associated with the selected statistics, such as forces, mass, velocity, acceleration, and position.
  - Mass: When the user ticks this checkbox, they will be able to observe the mass of the main object, which are provided by the user.
  - Velocity: When the user selects this checkbox, they will be able to view the velocity of the main object. The velocity represents the object's speed and direction of motion. The direction of the velocity is typically represented by a positive or negative sign
  - Acceleration: Enabling this checkbox will display the acceleration of the main object, which represents its rate of change in velocity. The direction of the acceleration is represented by a positive or negative sign
  - Position: This checkbox allows the user to monitor the position of the main object, indicating its location within the simulation. The position indicates the location of the object at any given point in time.

3 Design

3.1 General class diagram

In this project, we have chosen to implement the Model-View-Controller (MVC) architecture for its numerous advantages

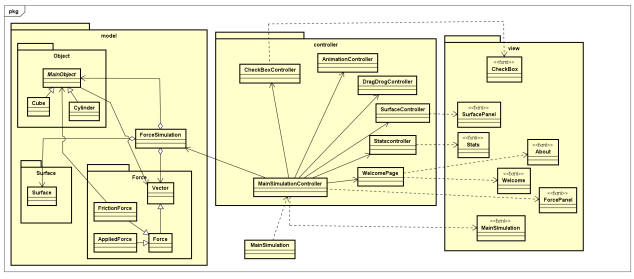


Figure 2: General class diagram

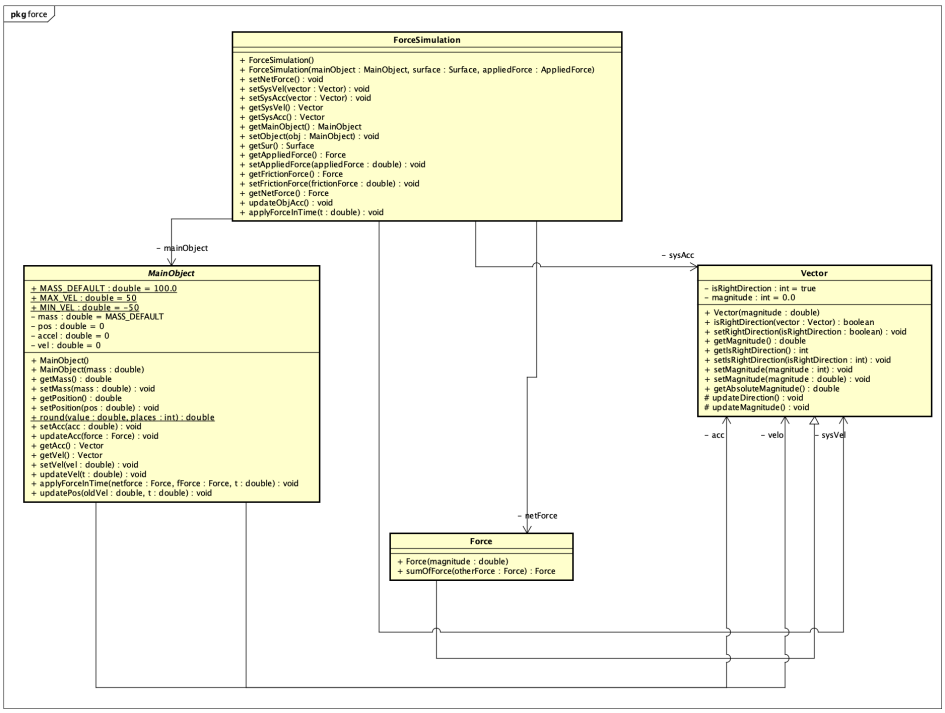


Figure 3: Model Diagram

3.2 Package details

3.2.1 Model Package

- Surface Package
  - Surface: The Surface class represents a surface and provides attributes and methods for adjusting the static and kinetic friction coefficients.
    - Constructor
      - Surface(): Default constructor for the Surface class.
      - Surface(double staticCoef): Constructor specifying the static friction coefficient. The kinetic friction coefficient is set to half of the static coefficient
      - Surface(double staticCoef, double kineticCoef): Constructor specifying both the static and kinetic friction coefficients.
    - Attribute
      - staticCoef: Represents the static friction coefficient of the surface.
      - kineticCoef: Represents the kinetic friction coefficient of the surface.
    - Method
      - getStaticCoef(): Returns the static friction coefficient of the surface.
      - getKineticCoef(): Returns the kinetic friction coefficient of the surface.
      - setStaticCoef(double staticCoef): Changes the static friction coefficient of the surface.
      - setKineticCoef(double kineticCoef): Changes the kinetic friction coefficient of the surface.

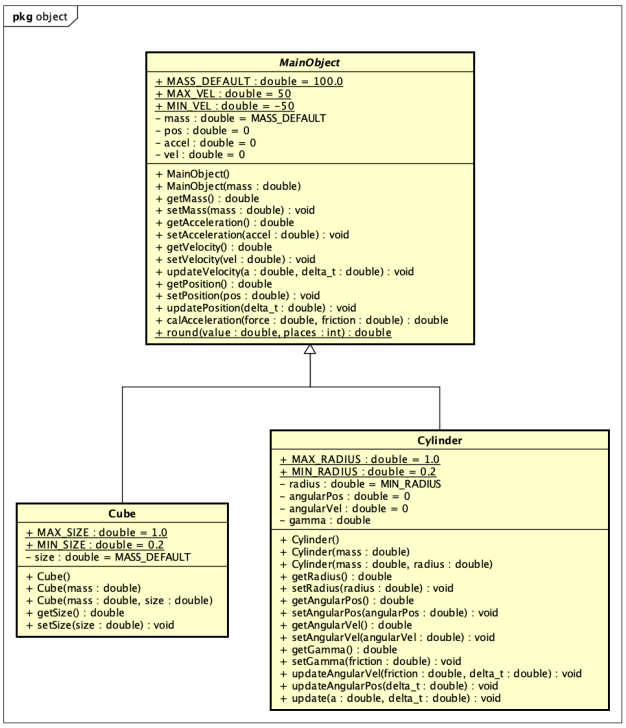


Figure 4: Object diagram

- Object Package
  - MainObject:
    - Contructor
      - MainObject(): Default constructor for the MainObject class.
      - MainObject(double mass): Constructor that accepts the mass parameter.
    - Constant
      - MASS_DEFAULT: Represents the default value of object's mass
      - MAX_VEL: Represents the maximum value of object's velocity
      - MIN_VEL: Represents the minimum value of object's velocity
    - Attribute
      - mass: Represents the mass of the main object. initially equals MASS_DEFAULT
      - pos: Represents the position of the main object. initially equals 0
      - acc: Represents the acceleration of the main object. initially equals 0
      - vel: Represents the velocity of the main object. initially equals 0
    - Method
      - getMass(): Returns the mass of the main object.
      - setMass(double mass): Sets the mass of the main object.
      - getAcceleration(): Returns the acceleration of the main object.
      - setAcceleration(double accel): Sets the acceleration of the main object.

        - updateAcceleration(double netForce): Updates the acceleration based on the net force applied.
        - getVelocity(): Returns the velocity of the main object.
        - setVelocity(double vel): Sets the velocity of the main object.
        - updateVelocity(double delta_t): Updates the velocity based on the current acceleration and a time interval.
        - getPosition(): Returns the position of the main object.
        - setPosition(double pos): Sets the position of the main object.
        - updatePosition(double delta_t): Updates the position based on the current velocity and a time interval.
        - applyForceInTime(Force netforce, Force fForce, double t): Applies the net force to the main object over a specified time interval, updating the acceleration, position, and velocity.
        - round(double value, int places): A static utility method to round a double value to a specified number of decimal places.
- Cube: The Cube class extends the MainObject class and represents a cube-shaped object.
    * Constructor
        - Cube(): Default constructor for the Cube class.
        - Cube(double mass): Constructor specifying the mass of the cube.
        - Cube(double mass, double size): Constructor specifying both the mass and size of the cube.
    * Constant
        - MAX_SIZE: Represents the maximum value of cube's size
        - MIN_SIZE: Represents the minimum value of cube's size
    * Attribute
        - size: Represents the size of the cube.
    * Method
        - getSize(): Returns the size of the cube.
        - setSize(double size): Sets the size of the cube, ensuring it is within the valid range specified by MAX_SIZE and MIN_SIZE.
- Cylinder: The Cylinder class extends the MainObject class and represents a cylinder-shaped object.
    * Constructor
        - Cylinder(): Default constructor for the Cylinder class.
        - Cylinder(double mass): Constructor specifying the mass of the cylinder.
        - Cylinder(double mass, double radius): Constructor specifying both the mass and radius of the cylinder.
    * Constant
        - MAX_RADIUS: Represents the maximum radius of the cylinder.
        - MIN_RADIUS: Represents the minimum radius of the cylinder.
    * Attribute
        - radius: Represents the radius of the cylinder.

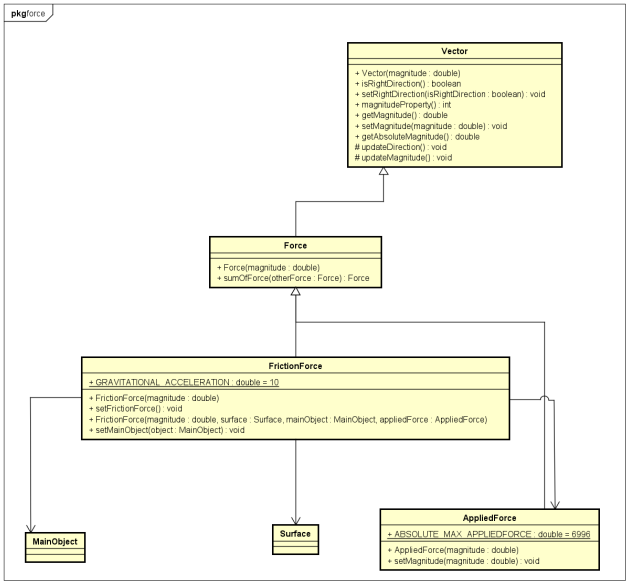


Figure 5: Force diagram

- angularPos: Represents the angular position of the cylinder.
- angularVel: Represents the angular velocity of the cylinder.
- gamma: Represents the friction coefficient gamma of the cylinder.

* Method
  - getRadius(): Returns the radius of the cylinder.
  - setRadius(double radius): Sets the radius of the cylinder, ensuring it is within the valid range specified by MAX_RADIUS and MIN_RADIUS.
  - getAngularPos(): Returns the angular position of the cylinder.
  - setAngularPos(double angularPos): Sets the angular position of the cylinder.
  - getAngularVel(): Returns the angular velocity of the cylinder.
  - setAngularVel(double angularVel): Sets the angular velocity of the cylinder.
  - getGamma(): Returns the friction coefficient gamma of the cylinder.
  - setGamma(double friction): Sets the friction coefficient gamma of the cylinder.
  - updateAngularVel(double friction, double delta_t): Updates the angular velocity of the cylinder based on the friction coefficient and time interval.
  - updateAngularPos(double delta_t): Updates the angular position of the cylinder based on the angular velocity and time interval.
  - applyForceInTime(Force netforce, Force fForce, double t): Applies the net force and frictional force to the cylinder over a specified time interval, updating the acceleration, position, velocity, angular position, and angular velocity of the cylinder.

- Force Package
  - Vector: The Vector class represents a vector quantity with a magnitude and a direction.
    * Constructor

        - Vector(double magnitude): Creates a Vector object with the specified magnitude.
    - Attribute
        - isRightDirection: Represents the direction of the vector. It is a BooleanProperty indicating whether the vector is in the right direction or not.
        - magnitude: Represents the magnitude of the vector. It is a DoubleProperty storing the magnitude value.
    - Method
        - isRightDirection(): Returns a boolean indicating whether the vector is in the right direction or not.
        - setRightDirection(boolean isRightDirection): Sets the direction of the vector to the specified value and updates the magnitude accordingly.
        - magnitudeProperty(): Returns the DoubleProperty object representing the magnitude.
        - getMagnitude(): Returns the current magnitude value of the vector.
        - setMagnitude(double magnitude): Sets the magnitude of the vector to the specified value and updates the direction accordingly.
        - getAbsoluteMagnitude(): Returns the absolute value of the magnitude.
        - updateDirection(): Updates the direction of the vector based on the magnitude value.
        - updateMagnitude(): Updates the magnitude of the vector based on the direction and absolute magnitude value.
- Force: The Force class extends the Vector class and represents a force vector.
    - Constructor
        - Force(double magnitude): Creates a Force object with the specified magnitude.
    - Method
        - sumOfForce(Force otherForce): Calculates the sum of two forces by adding their magnitudes and returns a new Force object representing the net force. The direction of the net force is updated accordingly.
- AppliedForce
    - Constructor
        - AppliedForce(double magnitude): Creates an AppliedForce object with the specified magnitude.
    - Constant
        - ABSOLUTE_MAX_APPLIEDFORCE: Represents the absolute maximum value for the magnitude of the applied force.
    - Method
        - setMagnitude(double magnitude): Overrides the setMagnitude method from the parent class to cap the magnitude value within the range of -ABSOLUTE_MAX_APPLIEDFORCE to ABSOLUTE_MAX_APPLIEDFORCE. This ensures that the magnitude of the applied force does not exceed the specified maximum value.
- FrictionForce: The FrictionForce class extends the Force class and represents a friction force vector.
    - Constructor

- FrictionForce(double magnitude): Creates a FrictionForce object with the specified magnitude.
- FrictionForce(double magnitude, Surface surface, MainObject mainObject, AppliedForce appliedForce): Creates a FrictionForce object with the specified magnitude, surface, main object, and applied force. The constructor automatically calculates and sets the friction force based on the provided parameters.

* Attribute
  - surface: Represents the surface on which the friction force is acting.
  - mainObject: Represents the main object on which the friction force is applied.
  - appliedForce: Represents the applied force acting on the main object.
  - GRAVITATIONAL_ACCELERATION: Represents the value of gravitational acceleration.
* Method
  - setFrictionForce(): Calculates and sets the magnitude of the friction force based on the properties of the main object, applied force, and surface. The direction of the friction force is determined based on the velocity and direction of the main object, as well as the magnitude and direction of the applied force.
  - setMainObject(MainObject object): Updates the main object and recalculates the friction force based on the new main object.

– ForceSimulation: The ForceSimulation class represents a simulation of forces acting on a main object.

* Constructor
  - ForceSimulation(): Creates a ForceSimulation object with default values for main object, surface, applied force, and friction force.
  - ForceSimulation(MainObject mainObject, Surface surface, AppliedForce appliedForce): Creates a ForceSimulation object with the specified main object, surface, and applied force. The constructor automatically calculates the friction force and net force based on the provided parameters.
* Attribute
  - mainObject: Represents the main object on which the forces are acting
  - surface: Represents the surface on which the main object is placed.
  - appliedForce: Represents the applied force acting on the main object.
  - frictionForce: Represents the friction force acting on the main object.
  - netForce: Represents the net force acting on the main object.
  - sysVel: Represents the system velocity.
  - sysAcc: Represents the system acceleration.
  - sysAngAcc: Represents the system angular acceleration.
  - sysAngVel: Represents the system angular velocity.
* Method
  - setNetForce(): Calculates and sets the net force based on the applied force and friction force.
  - setSysVel(Vector vector): Sets the system velocity.

- setSysAcc(Vector vector): Sets the system acceleration.
- getSysVel(): Returns the system velocity.
- getSysAcc(): Returns the system acceleration.
- getSysAngVel(): Returns the system angular velocity.
- getSysAngAcc(): Returns the system angular acceleration.
- setSysAngVel(double sysAngVel): Sets the system angular velocity.
- setSysAngAcc(double sysAngAcc): Sets the system angular acceleration.
- getMainObject(): Returns the main object.
- setObject(MainObject obj): Sets the main object and updates the system acceleration and velocity accordingly.
- getSur(): Returns the surface.
- setSur(Surface surface): Sets the surface and updates the friction force and net force accordingly.
- getAppliedForce(): Returns the applied force.
- setAppliedForce(double appliedForce): Sets the magnitude of the applied force and updates the friction force and net force accordingly.
- getFrictionForce(): Returns the friction force.
- setFrictionForce(): Updates the magnitude of the friction force based on the current state of the main object, applied force, and surface.
- getNetForce(): Returns the net force.
- applyForceInTime(double t): Applies the net force and friction force on the main object for a specified time duration.

3.2.2 Controller Package

- WelcomePage: The WelcomePage class is a controller class responsible for handling the welcome page of the application.
  - Attribute
    - gridPane: Represents the grid pane containing child nodes.
    - surface: Represents the image view of the surface.
    - cylinder: Represents the image view of the cylinder.
    - animationController: An instance of the AnimationController class.
  - Method
    - start(ActionEvent event): Handles the event when the user clicks the start button. It loads the main simulation scene from the FXML file and sets it as the new scene on the primary stage.
    - animation(TranslateTransition transition, double rate): Configures and starts the translation animation for a given transition and rate.
    - rotation(RotateTransition rotate, double rate): Configures and starts the rotation animation for a given rotation and rate.
    - initialize(URL location, ResourceBundle resources): Initializes the welcome page by setting up the animations for the grid pane, surface, and cylinder.

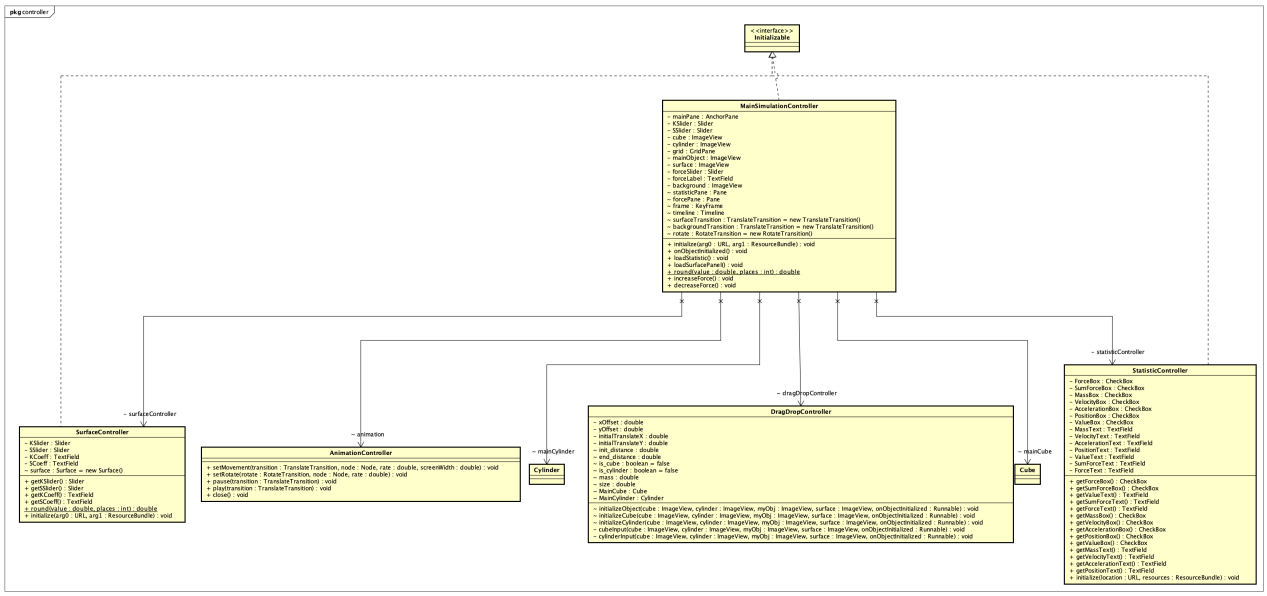


Figure 6: Enter Caption

- MainSimulationController: The MainSimulationController class is a controller class responsible for handling the main simulation scene of the application.
  - Attribute
    - mainPane: Represents the main anchor pane of the scene.
    - cube, cylinder, mainObject, surfaceView, surface, cloud1, cloud2, cloud3, cloud4: Represent the image views of various elements in the scene.
    - grid: Represents the grid pane containing child nodes.
    - forceSlider: Represents the slider for adjusting the applied force.
    - forceLabel: Represents the text field for displaying the value of the applied force.
    - increaseForce, decreaseForce: Buttons for increasing and decreasing the applied force.
    - surfaceController: An instance of the SurfaceController class.
    - checkBoxController: An instance of the CheckBoxController class.
    - statsController: An instance of the StatsController class.
    - animation: An instance of the AnimationController class.
    - dragDropController: An instance of the DragDropController class.
    - statisticPane: A pane for displaying statistics.
    - forcePane: A pane for controlling surface properties.
    - mainCube: An instance of the Cube class representing the main cube object.
    - mainCylinder: An instance of the Cylinder class representing the main cylinder object.
    - forceSimulation: An instance of the ForceSimulation class for simulating forces on the objects.
    - mainSurface: An instance of the Surface class representing the main surface.
    - appliedForce: An instance of the AppliedForce class representing the applied force on the object.

    - frictionForce: An instance of the FrictionForce class representing the friction force.
    - frame: A key frame for the animation timeline.
    - timeline: A timeline for animating the simulation.
    - surfaceTransition, cloudTransition1, cloudTransition2, cloudTransition3, cloudTransition4: Translate transitions for animating the surface and clouds.
    - rotate: A rotate transition for animating the rotation of the main object.
    - mass: A label for displaying the mass of the object.
  - Method
    - initialize(URL arg0, ResourceBundle arg1): Initializes the main simulation scene by setting up event handlers, loading UI components, and configuring animations.
    - loadStats(): Loads the statistics panel and adds it to the main pane.
    - onObjectInitialized(): Called when the main object is initialized (either cube or cylinder). Sets up the force simulation, animation, and updates the statistics.
    - showStats(): Updates and displays the statistics based on the current state of the main object.
    - disableForceController(boolean b): Disables or enables the force controller components based on the given boolean value.
    - loadCheckBox(): Loads the checkbox panel and adds it to the main pane.
    - loadSurfacePanel(): Loads the surface panel and adds it to the main pane.
    - round(double value, int places): Utility method for rounding a double value to a specified number of decimal places.
    - increaseForce(): Increases the applied force when the increase force button is clicked.
    - decreaseForce(): Decreases the applied force when the decrease force button is clicked.
    - reset(): Resets the simulation to its initial state by clearing the scene, resetting values, and enabling the selection of a new object.
    - play(): Resumes the animation and simulation.
    - pause(): Pauses the animation and simulation.

- AnimationController
  - Method
    - setTransition(Node node, double duration, double x_from, double x_to): Creates a translate transition for a given node with the specified duration and x-axis values for the starting and ending positions.
    - setMovement(TranslateTransition transition, Node node, double rate, double screenWidth): Configures a translate transition to move a node horizontally across the screen. The rate parameter controls the speed of movement, and the screenWidth parameter represents the width of the screen. The transition will loop indefinitely and reset the position of the node when it slides off the screen.
    - setRotate(RotateTransition rotate, Node node, double rate): Configures a rotate transition to rotate a node continuously. The rate parameter controls the speed of rotation. The transition will loop indefinitely and restart when it completes.

- DragDropController: The DragDropController class handles the dragging and dropping of objects in the application.
  - Method
    - initializeObject(ImageView cube, ImageView cylinder, ImageView myObj, ImageView surface, Runnable onObjectInitialized): Initializes the dragging and dropping functionality for both the cube and cylinder objects. It takes the necessary image views and a callback function onObjectInitialized that will be executed when an object is successfully dropped.
    - initializeCube(ImageView cube, ImageView cylinder, ImageView myObj, ImageView surface, Runnable onObjectInitialized): Initializes the dragging and dropping functionality for the cube object. It handles the mouse press, drag, and release events and sets the initial position and visibility of the cube.
    - initializeCylinder(ImageView cube, ImageView cylinder, ImageView myObj, ImageView surface, Runnable onObjectInitialized): Initializes the dragging and dropping functionality for the cylinder object. It handles the mouse press, drag, and release events and sets the initial position and visibility of the cylinder.
    - cubeInput(ImageView cube, ImageView cylinder, ImageView myObj, ImageView surface, Runnable onObjectInitialized): Displays a dialog for the user to input the properties of the cube object, such as mass and side length. It validates the input and creates a cube object based on the provided properties. It also sets the image, scale, and layout of the main object (myObj) and triggers the onObjectInitialized callback.
    - cylinderInput(ImageView cube, ImageView cylinder, ImageView myObj, ImageView surface, Runnable onObjectInitialized): Displays a dialog for the user to input the properties of the cylinder object, such as mass and radius. It validates the input and creates a cylinder object based on the provided properties. It also sets the image, scale, and layout of the main object (myObj) and triggers the onObjectInitialized callback.
- SurfaceController: The SurfaceController class is responsible for managing the surface properties in the application.
  - Method
    - initialize(URL arg0, ResourceBundle arg1): Initializes the surface properties by setting up event listeners for the kinetic and static coefficient sliders. It updates the surface object with the selected values and displays them in the corresponding text fields.
    - reset(): Resets the surface properties to their default values.
    - disableFrictionSlider(boolean b): Disables or enables the friction sliders based on the given boolean value.
- CheckBoxController: The CheckBoxController class is responsible for managing the statistic checkboxes in the application.
  - Method
    - initialize(URL location, ResourceBundle resources): Initializes the controller.
    - reset(): Resets all the checkboxes to their default state.

- StatsController: The StatsController class is responsible for managing the text fields that display the statistics of the object in the application.
  - Method
    - initialize(URL location, ResourceBundle resources): Initializes the controller.

3.2.3 View Package

In this project, we aim to break down the components into as many independent parts as possible to ensure convenience and ease of maintenance. We strive to minimize the impact of updates to one scene on others. The listed order of the FXML files represents the sequence in which we add scenes to the stage, ensuring a well-organized and structured management of the graphical components in the application.

- Welcome.fxml: The welcome scene features a dynamic background and a prominent "Start" button, creating an engaging and inviting atmosphere.
- MainSimulation.fxml :This scene is established as the container for the entire scene. Firstly, we create an Anchor Pane and incorporate the Sky and the Surface components. Following that, we generate multiple Rectangles to include the respective scenes.
- CheckBox.fxml: In this scene, we create a Pane and set its pointer for reference. Labels and checkboxes are added to the Pane to display and control the values such as force, velocity, and acceleration in the simulation. Finally, the Pane is added to the upper right of the Main Scene.
- ForcePanel.fxml: In this scene, we construct a Pane that includes an "Applied Force" label, a TextField for user input of the applied force value, and a Slider for adjusting the force. The Pane is then positioned in the bottom center of the Main Scene.
- Stats.fxml : In this scene, we generate a Statistic table to present parameters like Position, Velocity (including Angular Velocity), and Acceleration (including Angular Acceleration). The table is subsequently added to the upper left of the Main Scene.
- SurfacePanel.fxml: In this scene, we create a Pane that includes 2 TextFields and 2 Sliders to control the static and kinetic friction coefficients of the surface. These controls allow the user to adjust the friction coefficients as desired. The Pane is then added to the bottom right of the Main Scene.

4 References

References

[1] http://phet.colorado.edu/sims/html/forces-and-motion-basics/latest/forces-and-motion-basics_en.html("Acceleration" module)

[2] stackoverflow.com

[3] stackexchange.com

[4] https://youtu.be/9XJicRt_FaI