Hanoi University of Science and Technologies

School of Information and Communication Technology



IT3100E-131678

Object-oriented Programming

Mini-Project

Interactive simulation of the composition of forces

[Group 07]

Nguyen Quang Duc – 20204876

Nguyen The Minh Duc – 20204904

Nguyen Ngoc Dung – 20204905

Luu Anh Duc – 20204875

Ha Noi, July 14, 2022

Table of contents

[1 Assignment of members 1](#_Toc108814196)

[1.1 List of tasks 1](#_Toc108814197)

[2 Mini-project description 3](#_Toc108814198)

[2.1 Project overview 3](#_Toc108814199)

[2.2 Mini-project requirement 3](#_Toc108814200)

[2.3 Use case diagram 4](#_Toc108814201)

[2.3.1 Set up the main object 5](#_Toc108814202)

[2.3.2 Control the applied force 6](#_Toc108814203)

[2.3.3 Control the friction coefficients 6](#_Toc108814204)

[2.3.4 Choose to display detailed information 6](#_Toc108814205)

[2.3.5 Control simulation state 6](#_Toc108814206)

[3 Design 7](#_Toc108814207)

[3.1 General class diagram 7](#_Toc108814208)

[3.2 Package details 8](#_Toc108814209)

[3.2.1 Controller package 8](#_Toc108814210)

[3.2.2 Model package 22](#_Toc108814211)

[3.2.3 View 35](#_Toc108814212)

[4.References 37](#_Toc108814213)

# 1 Assignment of members

In this part, we will inform the work each group member has done by listing classes and methods where he mainly involved and then enumerate all materials which we based on in this project.

## List of tasks

|  |  |  |  |
| --- | --- | --- | --- |
| Nguyen Quang Duc  20204876 | Nguyen The Minh Duc  20204904 | Nguyen Ngoc Dung  20204905 | Luu Anh Duc  20204875 |
| Design general class diagram (34%)  Writing report (40%)  ObjectPanelController.java  StatisticPanelController.java  Cube.java  Cylinder.java  Rotatable.java | Design general class diagram (33%)  Design controller diagram  Writing report (50%)  ControlPanelController.java  ForcePanelController.java  ForceSimulationAppController.java  Simulation.java  Surface.java  StatisticsPanel.fxml  ControlPanel.fxml  ForcePanel.fxml  RootLayout.fxml  SurfacePanel.fxml | Design general class diagram (33%)  Design model diagram  Writing report (10%)  GameAnimationTimer.java  AnimationController.java  SurfacePanelController.java  MainObject.java  AppliedForce.java  Force.java  FrictionForce.java  HorizontalVector.java | Design use-case diagram  Design slide for presentation (100%)  Animation.fxml  ObjectPanel.fxml  Clean and comment codes.  Purpose idea and help other optimize their codes  Record demo video |

We break our project into four separate parts for the sake of assigning tasks for each member.

* Design
* General class diagram:
  + Nguyen Quang Duc 20204876 (34%)
  + Nguyen The Minh Duc 20204904 (33%)
  + Nguyen Ngoc Dung 20204905 (33%)
* Detail class diagram:
  + Model: Nguyen Ngoc Dung 20204905
  + Controller: Nguyen The Minh Duc 20204904
* Use-case diagram:
  + Luu Anh Duc 20204875
* Coding
  + Controller package:
    - Nguyen Quang Duc 20204876 (65%)
    - Nguyen The Minh Duc 20204904 (20%)
    - Nguyen Ngoc Dung 20204905 (15%)
  + Model package:
    - Nguyen Ngoc Dung (60%)
    - Nguyen The Minh Duc 20204904 (20%)
    - Nguyen Quang Duc 20204876 (20%)
  + View package:
    - Nguyen The Minh Duc 20204904 (70%)
    - Luu Anh Duc 20204875 (30%)
* Report:
  + Nguyen The Minh Duc 20204904 (50%)
  + Nguyen Quang Duc 20204876 (40%)
  + Nguyen Ngoc Dung 20204905 (10%)
* Slides
  + Luu Anh Duc 20204875 (100%)

# 2 Mini-project description

## 2.1 Project overview

* In this project, our group needs to create a simple interactive simulation application for demonstrating Newton’s laws of motion.
* All the works, we need to collaborate and share with each other by using Version Control, specifically on [Github](https://github.com/QuangDuc-HUST/OOP.DSAI.20212.07).
* To build this application successfully, we also need to construct a use-case diagram and class diagrams in the development process.
* Object-oriented programming concepts such as Inheritance, Polymorphism, and Association, etc. also must be applied into this project.
* All ideas for the projects, the reasons why we chose those also need to be explained in this report as well as our presentation.

## Mini-project requirement

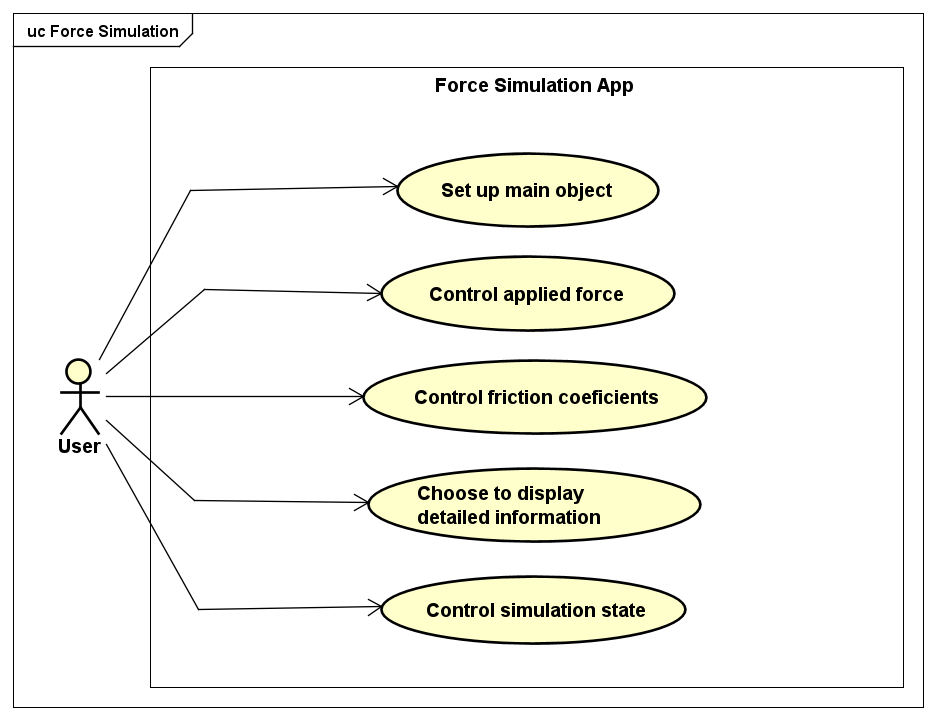
As we mentioned before, in this project, our group needs to build a simple interactive simulation application for demonstrating Newton’s laws of motion with some specific requirements:

* For the GUI, the program should look like in the reference [1] which includes two main parts, the sky and the surface.
* The user can control all the components of the physical system which includes one main object, the surface and an actor who can apply a horizontal force on the mass center of the object and then observe the motion of the main object.
* To start an application, the user needs to set up the main object by dragging an option (either a cube or a cylinder) from the object menu on the bottom left onto the surface and then specify suitable parameters based on the shape of the object:
  + Cube-shaped object: side-length, mass
  + Cylinder-shaped object: radius, mass
* While the simulation is running, the user can control:
  + The actor (which is represented by the force he applies) by changing the length and direction of the force in the bottom center panel by using the sliding bar or specify the number of Newtons in the textbox.
  + The static and kinetic friction coefficients of the surface by modifying its value in sliding bar or a text box in the bottom right panel.
  + All related statistics, which are all forces’ value and direction, sum of forces, mass, velocity, acceleration, and position of the main object (including angular units if the object is cylinder-shaped through corresponding tick-boxes on the panel on the upper right.
  + Pause, continue, and reset by simulation.
* To simulate motion, the program recalculates the statistics of the main object after each time interval Δt and how the physical force impacts on the object by the formular which the guidelines provided us.

Furthermore, there are some points we also need to clarify in our project beside some above requirements:

* + The user cannot choose another main object as we think it will make no sense when the user changes the cube-shaped object to cylinder-shaped object (what statistics, units we need to keep when change two different kinds of object?).
  + The user cannot modify any parameters of the current object (if we have time, we will update this functionality).
  + For two above functionalities, user can reset the application and choose the main object again.

## 2.3 Use case diagram



### 2.3.1 Set up the main object

* When setting up the main object: user need to choose its type (cylinder or cube) by “drag and drop”, and then pass value for object’s mass and size.
* After user pass in all parameters for object, our program will create modify and size of that object and display above the surface.
* The user can see the change in object’s size in accordance with the window size.

### 2.3.2 Control the applied force

* User can adjust the slider or just type value into the text box. If user type value into text box, he must hit enter to confirm this value.
* After user modified the applied force, our program will change the applied force to object and update the net force, accelerate, and the width of vector representing it.
* The user can see that the object is moving faster or slower, and the width of forces vector will be changed. The values of labels representing accelerate and force will be change according to the current applied force.

### 2.3.3 Control the friction coefficients

* Similar to the use case when user want to control the applied force, the user can adjust the slider or type value into the text box. If user type value into text box, he must hit enter to confirm this value.
* After user modified the coefficients of friction, our program will update the statistics such as changing the friction force and update the width of the vector representing that friction force.
* The user can see that the object is moving faster or slower, and values of labels representing accelerate and force will be changed according to the current coefficient frictions.

### 2.3.4 Choose to display detailed information

* User can choose to display the detailed information of forces, object by ticking the checkboxes.
* If user tick the checkboxes, our program will display values of respective label attributes.
* User can observe the value of these label if he checks the checkboxes. If he wants to not see the detailed information by unchecking the checkboxes. When observing the statistics, the user can have the knowledge about the physical system.

## 2.3.5 Control simulation state

* User can change simulation state such as start, pause, and continue by pressing “>” or “||” buttons and restart the project by “Reset” button
* When the user presses the buttons, the program will act corresponding to the meaning of the buttons by pause/continue the transition and rotation of the nodes or renew the stage.
* User can manipulate the stage of the program and have time to think about physical phenomenon by choosing to pause or create new physic system.

# 3 Design

## 3.1 General class diagram

In this project, we tried to apply MVC architecture for the sakes of its advantages and member assignment.

Diagram

Description automatically generated

## 3.2 Package details

### 3.2.1 Controller package

The general class diagram for our controller package is:

Diagram

Description automatically generated

First, we will break the controller into smaller components for convenient and maintenance purpose. The class ForceSimulationAppController is the main controller of our project so we could split its tasks to other controllers like AnimationController ObjectPanelController and ControlPaneController.

Diagram

Description automatically generated

* ForceSimulationAppController: This class is the controller of the file ‘RootLayout.fxml’ and is used for managing the main screen, control the reset and pause simulation button. It also is used to combine scenes to the stage in our program.
  + Its attributes are:
    - simul: our model from model package.
    - topStackPane: this attribute is used to contain 2 image views represent the above view for our simulation.
    - downStackPane: this attribute is used to contain 2 image views represent the under view for our simulation, and we also use it to control size of object.
    - controlPane: the GridPane to contain the controller for the friction force and applied force. It also use to contain the shape object represents our 2 objects (cube and cylinder).
    - aniController: this attribute is an instance of AnimationController class.
    - objController: this attribute is an instance of ObjectPanelController class.
    - controlController: this attribute is an instance of ControlPanelController class.
    - rootLayout: this attribute is instance of GridPane and it is used as a frame for the animation.
    - pauseButton: this attribute is instance of Button – which is used to pause or start the simulation.
    - resetButton: this attribute is instance of Button – which is used to reset the simulation.
    - brandLabel: this attribute is used to show our group ID.
  + Its methods are:
    - setSimul(Simulation simul): this method is used to pass the simulation to our simulation program.
    - init(Simulation simul): this method is used to pass the simulation to our program and also call 3 methods showAnimation(), showControlPane() and setUpPauseResetOperation() in order to initialize these 3 controllers.
    - showAnimation(): this method is used to set up the aniController of the AnimationController class.
    - showControlPane(): this method is used to set up the attribute controlController of the simulation program.
    - setUpPauseResetButton(): this method is used to set up the pause and reset button in the topStackPane and its behaviors during the simulation is running.
    - resetButtonPressed(): this method is used to reset the simulation program.
    - pauseButtonPressed(): this method is used to pause the simulation if it already started or it will continue or start the simulation program.
* AnimationController: This controller is used for controlling the rate which the topStackPane and downStackPane of out model move.
  + Its attributes are:
    - BACKGROUND\_WIDTH = 200: this is the default value for these 4 image views.
    - ratioTwoBackGround = 200.0f
    - TRANSITION\_DURATION = 15000
    - timer: this class is an instance of the GameAnimationTimer – which is used to handle each time frame during the simulation is running.
    - simul: this attribute is an instance of the Simulation class – which is our model.
    - parallelTransitionUp: this attribute is used to force the sky image to move in parallel with the ground.
    - parallelTransitionDown: this attribute is used to force the ground image to move in parallel with the sky.
    - topStackPane: this attribute is used to contain 2 image views (represented by the backGroundMiddleUp and backGroundRightUp) – which is set to move in parallel during the simulation. It is also used to contain labels and checkboxes controller for our program.
    - downStackPane: this attribute is used to contain 2 image views (represented by the backGroundMiddleDown and backGroundRightDown) – which is set to move in parallel during the simulation. It is also used to contain the 2 objects (Cube and Cylinder) and Sliders as well as TextField attributes to control the the applied force as well as the friction coefficients.
    - backGroundMiddleUp: this attribute represents the image view for the sky of the simulation.
    - backGroundRightUp: this attribute represents the image view for the sky of the simulation
    - backGroundMiddleDown: this attribute represents the image view for the ground of the simulation
    - backGroundRightDown: this attribute represents the image view for the ground of the simulation.
  + Its methods are:
    - getParallelTransitionUp(): this method is used to get the parallelTransitionUp – which is used to control the move of the sky of simulation.
    - getTopStackPane(): this method is used to get the topStackPane of the simulation.
    - getDownStackPane(): this method is used to get the downStackPane of the simulation.
    - init(Simulation sim): this method is used to set up the pointer of simul attribute to the same simul of our program – by the method setSim(Simulation simul) and also create the timer for the simulation.
    - setSim(Simulation simul): as describe above, this method is used to set the pointer of simul attribute to the same simul of our program.
    - startAnimation(): this method is used to start the simulation by starting the timer and calling method play() of the parallelTransitionUp and parallelTransitionDown.
    - continueAnimation(): this method is used to continue the simulation by starting the timer and calling method play() of the parallelTransitionUp and parallelTransitionDown.
    - pauseAnimation(): this method is used to pause the simulation by starting the simulation by pausing the timer and also calling method play() of the parallelTransitionUp and parallelTransitionDown.
    - resetAnimation(): this method is used to reset the simulation by stopping all the timer, parallelTransitionUp and parallelTransitionDown of the simulation and set the position of the parallelTransitionUp and parallelTransitionDown to the starting point.
* GameAnimationTimer: This class is an utility for our model – which is used for animation
  + Its attributes are:
    - lastFrameTimeNanos: this attribute is used to store the last time frame.
    - isPaused: this attribute is used to show that if the simulation model is paused.
    - pauseScheduled: this attribute is used to show that if the simulation will be paused.
    - playScheduled: this attribute is used to show that if the simulation will be played.
    - restartScheduled: this attribute is used to show that if the simulation will be restarted.
  + Its methods are:
    - pause(): this method is used to pause the simulation.
    - play(): this method is used to run the simulation.
    - start(): this method is used to run the simulation.
    - stop(): this method is used to stop the simulation.
    - handle(): as this class extends the AnimationTimer class, it need to implement handle method – which is abstract in the AnimationTimer class.
    - tick(float secondsSinceLastFrame): this is an abstract method, which will be used to update the accelerate, velocity and position base on the seconds from last frame.

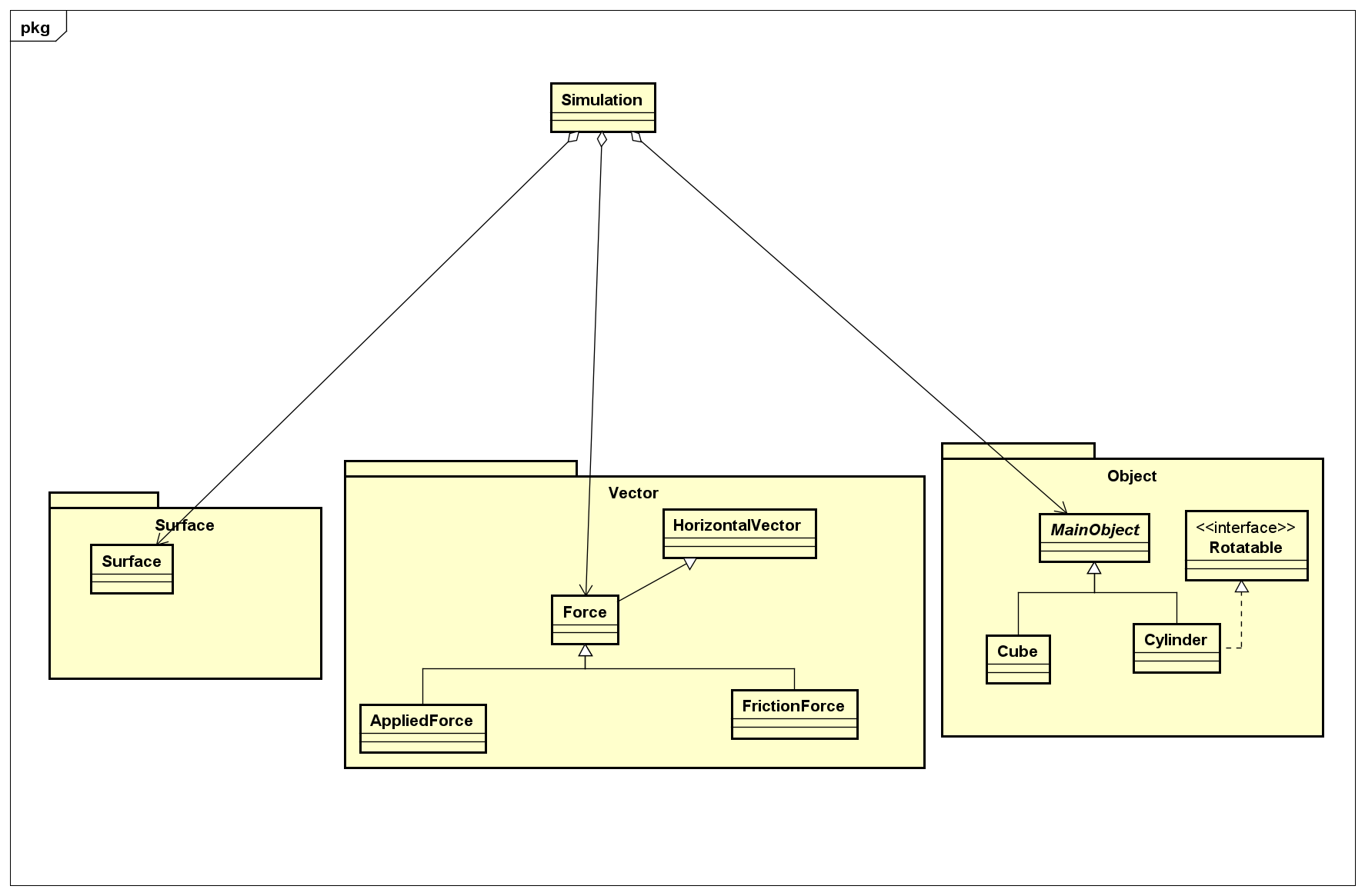
We split the task of ControlPanelController into the smaller controller class as we have to control the force, surface, object and also the statistical representation -> We create 4 smaller independent controller for them. Below is the class diagram for the control controller part in our project, because StatisticalPanelController has many attributes that have the same role, so we will hide them in the class diagram and list them when we explain this class in detail.

A picture containing diagram

Description automatically generated

* ControlPanelController: Its class is used to control some value for our simulation like force, friction coefficients force, object property, etc…
  + Its attributes are:
    - simul: this attribute is instance of Simulation class.
    - objController: this attribute is instance of ObjectPanelController class – which is used to control the object in our program.
    - staController: this attribute is instance of StatisticsPanelController class – which is used to control the statistics in our program.
    - forceController: this attribute is instance of ForcePanelController class – which is used to control the applied force in our program.
    - surfaceController: this attribute is instance of surfaceController class – which is used to control the friction coefficients of the surface in our program.
    - topStackPane: this attribute is instance of StackPane – which is used to contain 2 image views represent the sky of the simulation.
    - downStackPane: this attribute is instance of StackPane – which is used to contain 2 image views represent the ground of the simulation.
    - cir: this attribute is used to represent the Cylinder of the simulation if the main object in type of Cylinder.
    - rec: this attribute is used to represent the Cube of the simulation if the main object in type of Cube.
    - controlPanelGridPane: this attribute is used to control the controlPanelGridPane – which is used to contain the cylinder and cube object, the force panel to control the force applied to the main object and also the surface panel to control the coefficient of the friction force of the simulation.
  + Its methods are:
    - init(Simulation simul, StackPane topStackPane, StackPane downStackPane): this method is used to set the simul, topStackPane and downStackPane attribute of the simulation and call 4 method showObjectPanel(), showForcePanel(), showSurfacePanel() and showStatisticPanel() to set up all the control for the object, force, surface and statistic panel of the simulation.
    - setTopStackPane(StackPane topStackPane): this method get the topStackPane as the parameter and set its topStackPane attribute to the topStackPane.
    - setDownStackPane(StackPane downStackPane): similarly to the setTopStackPane() method, it will get the downStackPane as the parameter and set its downStackPane attribute to the downStackPane.
    - setSimul(Simulation simul): this method get an parameter instance of Simulation and then the simul parameter to this ‘simul’.
    - getObjController(): this method is used to get the objController attribute of the simulation.
    - getControlPanelGridPane(): this method is used to get the controlPanelGridPane of the simulation.
    - showObjectPane(): this method is used to set the objectPanelController and also the object panel controller of the simulation.
    - showStatisticsPane(): this method is used to set the statisticPanelController and also the statistics panel controller of the simulation.
    - showForcePane(): this method is used to set the forcePanelController and also the force panel controller of the simulation.
    - showSurfacePane(): this method is used to set the surfacePanelController and also the surface panel controller of the simulation.
* ObjectPanelController: Its class is used for controlling the input for the object’s mass, radius (if the object is a cylinder) or side length (if the object is a cube). This class is also used to control the rotation if the main object of type Rotatable (the object is Cylinder).
  + Its attributes are:
    - simul: this attribute is instance of Simulation class.
    - topStackPane: this attribute is instance of StackPane – which is used to contain 2 image views represent the sky of the simulation.
    - downStackPane: this attribute is instance of StackPane – which is used to contain 2 image views represent the ground of the simulation.
    - cirRotate: this attribute is instance of RotateTransition, is used for the cylinder to control the rotate.
    - girdPaneObjectContainer: this attribute is used to represent the gridPane of size (2x1) to contain the 2 objects (cylinder and cube).
    - rec: this attribute is used to represent the Cube object of the simulation.
    - cir: this attribute is used to represent the Cylinder object of the simulation.
  + Its methods are:
    - getRec(): this method is used to get the rectangle – which represents for the cube in simulation.
    - getCir(): this method is used to get the circle – which represents for the cylinder in simulation.
    - init(): this method is used to set up image for the cylinder and cube in simulation and set the pointer of the simul by calling the method setSimul(Simulation simul), topStackPane and downStackPane to the same simul, topStackPane, downStackPane of the Simulation model. Then we call 2 methods setUpDragAndDrop() and setUpCircleRotation() – which will be explained latter.
    - setSimul(Simulation simul): this method is used to set the pointer of the simul attribute in this class to the same simul in our program.
    - setDownStackPane(StackPane downStackPane): this method is used to set the pointer of the attribute downStackPane to the same downStackPane of the model, and then we will bind the Radius Property of the Cylinder, Height and Width Properties of the Cube to the 30% height of the downStackPane.
    - setTopStackPane(): this method is used to set the pointer of the.
    - resetObjectPosition(): this method is used to reset all the object in the simulation, which mean we will set the size and mass of the object to the default value as well as move it to the gridPaneObjectContainer.
    - setUpCircleRotation(): this method is used to set up the parameter for the Rotation of Circle, and then if the object of the Simulation model is actually Cylinder, then we will bind the angular values of this cylinder to the rate of the Rotation.
    - startCirAnimation(): this method is used to start the rotation of the cylinder in our simulation model.
    - continueCirAnimation(): this method is used to continue the rotation of the cylinder in our simulation model.
    - pauseCirAnimation(): this method is used to pause the rotation of the cylinder in our simulation model.
    - resetCirAnimation(): this method is used to reset the rotation of the cylinder in our simulation model by set its position to the starting point and also stop rotate.
    - cubeInput(): this method is used to create an Alert dialog for user to type in the mass and size length of the cube, and then this method will update the size and mass of the object base on the inputted value.
    - cylinderInput(): this method is used to create an Alert dialog for user to type in the mass and radius of the cylinder, and then this method will update the radius and mass of the object base on the inputted value.
    - setUpDragAndDrop(): this method is used to set up actions for drag and drop object. If the simulation is running, we don’t allow to drag the object.
  + Its private class:
    - EventDragDetected: this class is used to catch event when we pressed the cursor on the object. It will create an overlay over object to drag it.
* StatisticsPanelController: This class is used for controlling the representation of the property of our model like applied force, friction force, object’ properties as well as its velocities and accelerates.
  + Its attributes are:
    - simul: this attribute is instance of Simulation class and used to represents the simulation in our model.
    - stackPane: this attribute is used to contain Checkboxes and Labels representing respective statistics.
    - rec: this attribute is used to represent the cylinder of the simulation.
    - cir: this attribute is used to represent the cube of the simulation.
    - aArrow: this attribute is used to contain vector image represent for applied force.
    - fArrow: this attribute is used to contain vector image represent for friction force.
    - nArrow: this attribute is used to contain vector image represent for net force.
    - angLabel: this Label attribute is used to show the value of current angle.
    - angAccLabel: this Label attribute is used to show the value of the current angular accelerate.
    - angVelLabel: this Label attribute is used to show the value of the current angular velocity.
    - posLabel: this Label attribute is used to show the value of the current position.
    - accLabel: this Label attribute is used to show the value of the current accelerate.
    - velLabel: this Label attribute is used to show the value of the current velocity.
    - aForceLabel: this Label attribute is used to show the value of the current applied force.
    - fForceLabel: this Label attribute is used to show the value of the current friction force.
    - sumForceLabel: this Label attribute is used to show the value of the current net force.
    - posCheckBox: this CheckBox attribute is used to control the visibility of the posLabel attribute.
    - accCheckBox: this CheckBox attribute is used to control the visibility of the accLabel attribute.
    - velCheckBox: this CheckBox attribute is used to control the visibility of the velLable attribute.
    - forceCheckBox: this CheckBox attribute is used to control the visibilities of the aArrow and fArrow attributes.
    - sumForceCheckBox: this CheckBox attribute is used to control the visibility of nArrow attribute.
    - masCheckBox: this CheckBox attribute is used to control the visibility of the massLabel attribute.
    - massLabel: this attribute is used to show the mass of the object.
    - valueCheckBox: this attribute is used to control the visibilities of the aArrowLabel, fArrowLabel and nArrowLabel.
    - aArrowLabel: this attribute is used to show the value of the current applied force.
    - fArrowLabel: this attribute is used to show the value of the current friction force.
    - nArrowLabel: this attribute is used to show the value of the current net force.
  + Its methods are:
    - init(): this method is used we set up the initial text value for the some Label attributes and set up the pointer of the cir and rec attributes to the same cir and rec of the simulation model. We also call method setSimul(), setTopStackPane(), setUpAppliedForce(), setUpFrictionForce(), setUpNetForce() – which will be explained in detail latter.
    - setSimul(Simulation simul): this method is used to set the pointer of the simul attribute to the same simul of the Simulation. Then, we will bind the visible properties of the aArrowLabel and fArrowLabel with the selected property of forceCheckBox attribute, the visibility of the nArrowLabel with the selected property of sumForcesCheckBox attribute. The visible properties of aForceLabel, fForceLabel and sumForceLabel is similar to the visible properties of the associated arrows but they are bound with the selected properties of the valueCheckBox attribute. The visible properties of other Labels will be bound with its associated Checkbox attributes. Finally, base on the type of object in simulation model, we will bind the visible properties of Label attributes with associated CheckBox attributes.
    - setUpNetForce(): this method is used to create and set the starting point of nArrow in the middle of the object.
    - setUpFrictionForce(): this method is used to create and set the starting point of fArrow in the middle of the object.
    - setUpAppliedForce(): this method is used to create and set the starting point of aArrow in the middle of the object.
    - setTopStackPane(StackPane topStackPane): this method is used to set the pointer of the stackPane point to the same topStackPane of the Simulation program, and then set up the position for label and representing statics value.
* ForcePanelController: This class is used to control the applied force, friction force coefficients and net force of our model
  + Its attributes are:
    - simul: this attribute is instance of Simulation class.
    - forcePanel: this StackPane is used to contain the forceSlider and forceTextField attributes.
    - forceSlider: this Slider is used to control the applied force on the object, you can adjust the slider to change the value for the applied force.s
    - forceTextField: this Text Field is used to control the applied force on the object, you can type value between the valid range to the text field to change the applied force.
  + Its methods are:
    - initialize(): this method is used to disable the forceTextField and forceSlider. Then we check validate text.
    - init(Simulation simul): this method is used to set the pointer of the simul to the same simul in the Simulation model. Then we will bind forceTextField and forceSlider with each other. End base on values of the applied force and friction force, we will update value for 3 forces. Finally we will check if the object is null – which means that the model doesn’t contain object, so we don’t allow user to adjust value of the slider and text field.
    - forceTextFieldOnAction(ActionEvent event): this method is called whenever we hit enter on force text field, then we will check for the validate text and range of the input, then we will update the applied force of the simulation base on that value.
    - netForceListener(): this method is used to set up listener for the change of the applied force and friction force, we will update the net force base on these changes.
* SurfacePanelController: This class is used to control the property of surface in our model like static friction coefficients and kinetic friction coefficients
  + Its attributes are:
    - simul: this attribute is instance of Simulation class.
    - staticCoefTextField: this attribute is used to control the value of the static coefficient friction force of the surface.
    - kineticCoefTextField: this attribute is used to control the value of the kinetic coefficient friction force of the surface.
    - staticCoefSlider: this attribute is used to control the value of the static coefficient friction force of the surface.
    - kineticCoefSlider: this attribute is used to control the value of the kinetic coefficient friction force of the surface.
  + Its methods are:
    - initialize(): this method is used to bind values of Text Field with the associated Slider attributes – which control the static coefficient and kinetic coefficient. We also need to check for validity of value we type in the Text Field
    - init(Simulation simul): this method is used to bind values of Text Field with the associated Slider attributes – which control the static coefficient and kinetic coefficient. We also check for the text validity by using try catch block, if values user typed in is not valid, then we will throw alert for that invalid value. Finally, we call the method surfaceListener() in order to start the listener to coefficient of surface.
    - surfaceListener(): this method is use to set up for the listener to catch up any changes in values of the static and kinetic coefficients, and then update the value of the friction force base on these values.
    - staticTextFieldOnAction(ActionEvent event): this method is used to get value for the static coefficient of surface in our program, it is also used to check that if the value user passed in is in the valid range or not.
    - kineticTextFieldOnAction(ActionEvent event): this method is used to get value for the kinetic coefficient of surface in our program, it is also used to check that if the value user passed in is in the valid range or not.

### 3.2.2 Model package



* Simulation: this is our model – which uses for logic to control surface, object and the force in the simulation.
  + Its attributes are:
    - isStart: This is a boolean property to represent that do the simulation is already start.
    - isPause: This is a boolean property to represent that do the simulation is stopped.
    - obj: The main object in our model (which can be cylinder or cube).
    - sysVel: Velocity of the object when the simulation starts.
    - sysAcc: Accelerate of the object when the simulation starts.
    - sysAngAcc: Angular accelerate of the object when the simulation starts.
    - sysAngVel: Angular velocity of the object when the simulation starts.
    - netForce: The net force of the simulation.
    - surface: This object is used to represent the surface in our simulation.
    - aForce: This object is used to represent the applied force in our simulation.
    - fForce: This object is used to represent the friction force in our simulation.
  + Its constructors are:
    - Simulation(): This constructor initialize of model with have surface, Applied Force = 0 (N) and Friction Force = 0 (N).
    - Simulation(MainObject mainObj, Surface surface, AppliedForce aForce): We pass the applied force, surface and object (which can be cube or cylinder) to this method. All these attributes affect the friction force. Inside this method, after setting the applied force, object and surface, we create the friction force for this model and also update the net force base by the method updateNetForce() base on the applied force and friction force.
  + Its methods are:
    - setSysVel(HorizontalVector horizontalVector): it is used to set the velocity of the object during the simulation process.
    - setSysAcc(HorizontalVector horizontalVector): similarly to the setSysVel(), this method is also used to set the velocity of the object during the simulation process.
    - sysVelProperty(): this method is used to get the vector velocity property of the object.
    - sysAccProperty(): this method is used to get the vector accelerate property of the object.
    - getSysAngAcc(): this method is used to get the angular accelerate property of the object if the object in type of cylinder.
    - setSysAngAcc(double sysAngAcc): this method is used to set the value for the angular accelerate property of the object if the object in type of cylinder.
    - getSysAngVel(): this method is used to get the angular velocity property of the cylinder object.
    - setSysAngVel(double sysAngVel): this method is used to set the value for the angular velocity property of the cylinder object.
    - objProperty(): this method is used to get the Object Property.
    - setObject(MainObject obj): this method is used to set the object (which can be cylinder or cube) for the model.
    - getObj(): this method will return the object in model.
    - isPauseProperty(): this method is used to get the Pause Boolean Property of the simulation.
    - getIsPause(): this method is used to get the boolean value of the isPause attribute of the simulation.
    - setIsPause(boolean isPause): this method is used to set the boolean value for the Pause Property of the model.
    - isStartProperty(): this method is used to get the Start Boolean Property of the simulation.
    - getIsStart(): this method is used to get the boolean value of the isStart attribute of the simulation.
    - setIsStart(boolean isStart): this method is used to set the boolean value for the Start Property of the model.
    - getSur(): this method is used to get the surface of the model.
    - getaForce(): this method is used to get the applied force of the model.
    - setaForce(double aForce): this method is used to set the value for applied force of the model.
    - getfForce(): this method is used to get the friction force of the model.
    - getNetForce(): this method is used to get the net force of the model.
    - start(): this method is used to start the simulation by set the isStart to True and isPause to False.
    - pause(): this method is used to pause the simulation by set the isStart to True.
    - conti(): this method is used to continue the simulation by set the isPause to False. We need to set name for this method to ‘conti’ instead of ‘continue’ as ‘continue’ is the reserved word in java.
    - restart(): this method is used to reset all Properties of the Simulation to the initial value.
    - updateObjAcc(): this method is used to update the accelerate of the object in model.
    - getObjVel(): this object is used to get the vector velocity property of the object.
    - updateNetForce(): this method is used to update the net force applied to the object by calling the static sumTwoForce – which combine the value of the applied force and friction force, then set that combined value to the net force.
    - applyForceInTime(double t): this method is used to update the accelerate, velocity and also the position of the object during the simulation. If the object is an instance of Cylinder, then we will also update the angular accelerate, angular velocity and the angle of the object.

#### 3.2.2.1 Object

* MainObject: This class is an abstract class, which is the base class for 2 objects cylinder and cube class. It contains the common property and also method for the object in simulation like mass, velocity, accelerate
  + Its constructors are:
    - MainObject(): if we do not pass any value for the object’s mass then the object will get 10kg mass for default.
    - MainObject(double mass): we set the mass of the object to this value.
  + Its attributes are:
    - mass: this attribute is in type of double property – which has default value is 10 kg.
    - acc: this attribute represents the accelerate in type HorizontalVector of the object and has default value is 0 m/s.
    - vel: this attribute represents the velocity in type HorizontalVector of the object and has default value is 0 m/s2.
    - pos: this attribute represents the position of the object relative to the starting point after the simulation is started.
  + Its methods are:
    - massProperty(): this method is used to get the mass of the object in type of property.
    - getMass(): this method is used to get the value of the mass of the object.
    - setMass(double mass): this method is used to set the mass value for the object’s mass. We check that if the mass is not larger than 0 than we will not except this value, and then we throw an Exception.
    - accProperty(): this method is used to get the vector accelerate property of the object.
    - velProperty(): this method is used to get the vector velocity property of the object.
    - setAcc(double acc): this method is used to set the value for the vector accelerate property of the object.
    - setVel(double vel): this method is used to set the value for the velocity accelerate property of the object.
    - updateAcc(Force force): this method is used to update the accelerate of the object base on force and object’s mass.
    - updateVel(double t): this method is used to update the velocity of the object base on it’s current velocity and accelerate. We also need to check that if the velocity decreases non-positive value then we will set it to 0.
    - applyForceInTime(Force netForce, Force force, double t): This method is used update the accelerate, velocity and also the position of the object base on the applied force, time and also its current velocity.
    - posProperty(): this method is used to get the position property of the object relative to its starting point.
    - getPos(): this method is used to get the position value of the object relative to its starting point.
    - setPos(double pos): this method is used to set the value for the position of the object.
    - updatePos(double oldVel, double t): this method is used to update the position of the object base on its old velocity, position and also its accelerate as well as time.
* Rotatable: This is an interface which is used for Cylinder to represents the rotate for cylinder
  + Its methods are:
    - angAccProperty(): this method is used to get the property value for the angular accelerate if the cylinder object.
    - getAngAcc(): this method is used to get the value for the angular accelerate of the cylinder object.
    - setAngAcc(double angAcc): this method is used to set the value for the angular accelerate of the cylinder object.
    - updateAngAcc(Force force): this method is used to update the angular accelerate of the cylinder object base on its friction force. If the type of force we pass in this method is not in friction force then we will throw an exception.
    - angVelProperty(): this method is used to get the value property for the angular velocity of the cylinder object.
    - getAngVel(): this method is used to get the value for the angular velocity of the cylinder object.
    - setAngVel(double angel): this method is used to set the value to the angular velocity property of the cylinder object.
    - updateAngVel(double t): this method is used to update the angular velocity of the cylinder object.
    - angleProperty(): this method is used to get the angular property of the cylinder object.
    - getAngle(): this method is used to the value of the current angle of the cylinder object.
    - setAngle(double angle): this method is used to set value for the angle of the cylinder object.
    - updateAngle(double oldAngVel, double t): this method is used to set the value for the angle of the cylinder object base on its old angular velocity and time it has passed when the simulation is started.
    - radiusProperty(): this method is used to get the value property of the radius of the cylinder.
    - getRadius(): this method is used to get the value of the radius of the cylinder.
    - setRadius(double radius): this method is used to set the value for the radius of the cylinder. When setting the value for radius, we need to check that if the value is in the valid range, if this value is invalid, we will throw the Exception.
    - applyForceInTimeRotate(Force force, double t): this method is used similarly to the applyForceInTime method except its update the value for the angular attributes.
* Cube: This class inherits from the MainObject class and add some attributes as well as methods from MainObject class in order to represents the cube object in our simulation model.
  + Its constructor:
    - Cube(): this method is inherited from the base class MainObject, and then it will set the value for size length to default value.
    - Cube(double mass): this method is inherited from the base class MainObject, and then it will set the value for size length to default vaue.
    - Cube(double mass, double size): this method will set value for mass and size of the object.
  + Its attributes:
    - size: this attribute is the side length of the cube and it has default value to be 0.3
    - MAX\_SIZE = 1.0: maximal value in the valid range for the side length of cube.
    - MIN\_SIZE = 0.1: minimal value in the valid range for the side length of cube.
  + Its methods:
    - sizeProperty(): this method is used to get the value property of the size of the object.
    - getSize(): this method is used to get the value of the size of the object.
    - setSize(double size): this method is used to set value for the the size of the object.
* Cylinder: This class inherits from the MainObject class and add some attributes as well as methods from MainOject class in order to represents the cylinder object in our simulation model.
  + Its constructor:
    - Cylinder(): this method is inherited from the base class MainObject and then it will set the value for the radius to the default value.
    - Cylinder(double mass): this method is inherited from the base class MainObject and set the value for the radius to the default value.
    - Cylinder(double mass, double radius): this method is use to set the values for the mass and radius object.
  + Its attributes:
    - angle: this attribute represents the double property angle of the object.
    - angAcc: this attribute represents the double property accelerate of the object.
    - angVel: this attribute represents the double property velocity of the object.
    - radius: this attribute represents the double property radius of the object and it has default value to be 0.3
    - MAX\_RADIUS = 1.0 :maximal value in the valid range for the radius.
    - MIN\_RADIUS = 0.1 : minimal value in the valid range for the radius
  + Its methods:
    - angAccProperty(): this method is used to get the double property value of accelerate of the cylinder.
    - getAngAcc(): this method is used to get the value of the accelerate of the cylinder.
    - setAngAcc(double angAcc): this method is used to set the value of the accelerate of the cylinder.
    - updateAngAcc(Force force): this method is used to update the value of the angular accelerate base on its friction force.
    - angVelProperty(): this method is used to get the double value of the angular velocity of the cylinder.
    - getAngVel(): this method is used to get the value of the angular velocity of the cylinder.
    - setAngVel(double angVel): this method is used to set value for the angular velocity of the cylinder.
    - updateAngVel(double t): this method is used to update the value of the angular velocity of the cylinder object base on time has passed.
    - angleProperty(): this method is used to get the property value of the angular of the cylinder object.
    - getAngle(): this method is used to get the value of the angle of the cylinder object.
    - setAngle(double angle): this method is used to set the value of the angle of the cylinder object.
    - updateAngle(double oldAngle, double t): this method is used to update the value of the angle of the cylinder object base one the old angle and time it has passed.
    - radiusProperty(): this method is used to get the value property of radius of the cylinder.
    - getRadius(): this methos is used to get the value of radius of the cylinder.
    - setRadius(double radius): this method is used to set the value for the radius of the cylinder. We also check for the validity of the value pass into this method
    - applyForceInTimeRotate(Force force, double t): this method is used to update the angular accelerate, angular velocity and also the angle base on the current net force and also time has passed.
    - applyForceInTime(Force force, double t): this method call both the method applyForceInTime(Force netForce, Force force, double t) from MainObject class and also applyForceInTimeRotate() to update value for the positions, velocities and accelerates of the Cylinder object.

#### 3.2.2.2 Surface

* Surface: This class represents the surface in our model – which is use for logic for surface’s properties
  + Its constructors:
    - Surface(): this constructor will initialize a surface with default values.
    - Surface(double staCoef): this constructor will create a new surface object and set the static coefficient to staCoef while the kinetic coefficient to staCoef /2 .
    - Surface(double staCoef, double kiCoef): this constructor will create a new surface object and set the static coefficient to staCoef and the kinetic coefficient to kiCoef.
  + Its attributes are:
    - staCoef: this attribute represents the value property of the static coefficient of the surface.
    - kiCoef: this attribute represents the value property of the kinetic coefficient of the surface.
    - MAX\_STA\_COEF: this attribute represents the maximum value for the static coefficient and also the kinetic coefficient of the surface.
    - STEP\_COEF: this attribute is the minimal scale for friction coefficients of the surface.
  + Its methods are:
    - staCoefProperty(): this method is used to get the value property of the static coefficient of the surface.
    - getStaCoef(): this method is used to get the value of the static coefficient of the surface.
    - kiCoefProperty(): this method is used to get the value property of the kinetic coefficient of the surface.
    - getKiCoef(): this method is used to get the value of the kinetic coefficient of the surface.
    - setStaCoef(double staCoef): this method is used to set the value for the static coefficient friction force of the surface.
    - setKiCoef(double kiCoef): this method is used to set the value for the kinetic coefficient friction force of the surface.

#### 3.2.2.3 Vector

* HorizontalVector: This class is used to represent the vector
  + Its constructor is:
    - HorizontalVector(double value): this constructor will set the value for the value attribute of this vector.
  + Its attributes are:
    - direction: this attribute uses boolean property to represent direction of this vector. If the value is true, then this vector will point in the positive direction (which is right in our simulation), otherwise it will point in the negative direction.
    - value: this attribute represents value property of the vector.
  + Its methods are:
    - directionProperty(): this method is used to get boolean property of the direction of vector.
    - getDirection(): this method is used to get boolean value of the direction of vector.
    - setDirection(boolean isRight): this method is used to set boolean property value of the direction.
    - valueProperty(): this method is used to get value property of vector.
    - getValue(): this method is used to get value of vector.
    - setValue(double value): this method is used to set value for the attribute ‘value’ of vector.
    - getLength(): this method is used to get absolute value of vector (ignore its sign).
    - updateValueDirection(): this method is used to update the direction of the vector to true if it points to the positive direction (which is right) or false if it points to the negative direction.
    - updateDirectionValue(): this method is used to update the value of the vector to positive if it points to the right direction or negative if it points to the left direction.
* Force: This class is used to represent the force
  + Its constructor is:
    - Force(double value): this constructor inherits from its base class – which means it will set the value to the double property value of vector – which is inherited from the HorizontalVector.
    - sumTwoForce(Force f1, Force f2): static method: this method is used to combine 2 force. It will combine value of f1 and f2, and then bases on this value, it will create a new Net Force of the simulation.
* FrictionForce: This class extends the Force class and is used to represents the friction force in our simulation model
  + Its constructors are:
    - FrictionForce(double value): this constructor is used to create a new friction force and set its value to value.
    - FrictionForce(double value, Surface surface, MainObject mainObj, AppliedForce aForce): this constructor is used to set the surface, main object and applied force to surface, mainObj and aForce, respectively. And then, base on the type of the main object, it will update the friction force accordingly.
  + Its attributes are:
    - surface: this attribute is used to represent the surface in the simulation model.
    - mainObj: this attribute is used to represent the object in the simulation model.
    - aForce: this attribute is used to represent the applied force in the simulation model.
    - g = 10: gravitational acceleration value (default value)
  + Its methods are:
    - updateFrictionForce(): this method is used to update the friction force base on the type of main object in the simulation.
    - setMainObj(MainObject obj): this method is used to set the object of the simulation to the obj.
* AppliedForce: This class extends the Force class and is used to represents the applied force in our simulation model
  + Its constructor is:
    - AppliedForce(double value): this constructor is used to create a new applied force and then use the constructor from the Force class to set the value for the applied force.
  + Its method is:
    - setValue(double value): this method is used to set the value for the applied force, if the applied force is not in the valid range, then it will set value of the applied force to the default value to its maximum or minimum value, otherwise, it will set value of the applied force to the value.

### 3.2.3 View

In this project, as there are many components so we want to break it to as many independent as we can for convenient and maintenance purpose, so that when we want to update some things from 1 scene, it will not affect others. The order that we list the fxml file below is the order that we add scenes to the stage.

Diagram

Description automatically generated

* RootLayout.fxml: In the RootLayout, we create 2 GridPane of size (1x2) for the topStackPane and downStackPane components. Then we add an AnchorPane that contains the reset and play button to the Pane(0,0).
* Animation.fxml: After set up layout for the simulation, we add image to the Scene by add 2 StackPane that contain image views represent the sky and ground. On topStackPane, we add 2 similar image views represent the sky and also 2 similar image views on downStackPane represent the ground.
* ControlPane.fxml: Then, in ControlPane.fxml, we create a new GridPane (3x1) for the control object, force and surface purposes. We add this to the downStackPane, above 2 images as we don’t need to interact with image views anymore, just control its rate when simulation is running.
* ObjectPanel.fxml: Here, we create a GridPane of size (2x1) to contain Cube and Cylinder objects, respectively and then add this GridPane to the downStackPane at index (0,0).
* StatisticsPanel.fxml: In this scene, we create a StackPane and set the pointer of that StackPane to the same topStackPane of rootLayout. Then, we add labels and checkboxes – which control the representations of values of force, velocity, accelerate, … in the simulation.
* ForcePanel.fxml: Here, we create a GridPane (3x1) – which contains a Force label, a TextField for user to type in value for the applied force and a Slider for user to adjust force. Then, we add this GridPane to the downStackPane at index (1,0).
* SurfacePanel.fxml: Finally, to control the static and kinetic friction coefficient of the surface, we create a GridPane of size (3x3) – which contains 2 TextField and Slider to adjust the friction coefficients of the surface, and add this GridPane to the downStackPane at index (2,0).

# 4.References

[1] <http://phet.colorado.edu/sims/html/forces-and-motion-basics/latest/forces-and-motion-basics_en.html>

Our main source for this project, our idea for GUI comes from the website.

[2] <https://edencoding.com/>

The website provides us some knowledge about MVC architecture, Animation Timer, text validation, drag and drop, Alert dialog, …

[3] <https://bekwam.blogspot.com/>

The blog supports us in Transition (background looping), Rotation (cylinder) of the project and MVC examples.

[4] <https://docs.oracle.com/javase/8/javafx/api/toc.htm>

Oracle API for JavaFX with numerous numbers of examples

[5] <https://stackoverflow.com/>

[6] <https://bugs.openjdk.org/>

We found the solution for our bugs in those forums.