Hanoi University of Science and Technologies

School of Information and Communication Technology



IT3103-143577

Object-oriented Programming

Mini-Project

Interactive simulation of the composition of forces

[Group 24]

Nguyen Trong Nghia – 20215101

Nguyen Van Nam – 20215097

Pham Thanh Nam – 20194630

Le Duy Nghiem – 20215102

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

Trong phần này, nhóm sẽ thông báo về công việc mà mỗi thành viên nhóm đã thực hiện bằng cách liệt kê các lớp và phương thức mà họ chủ yếu tham gia, sau đó liệt kê tất cả các tài liệu mà nhóm đã dựa trên trong dự án này.

## List of tasks

|  |  |  |  |
| --- | --- | --- | --- |
| Nguyen Trong Nghia | Nguyen Van Nam | Pham Thanh Nam | Le Duy Nghiem |
| General class diagram  General controller diagram  Writing report  Design slide for presentation  Skeleton code  ForceSimulationApp.java  Simulation.java  ControlPanelController.java  ForceSimulationAppController.java  tests | Detail controller diagram  Rotatable.java  MainObject.java  Force.java  FrictionForce.java  AnimationController.java  GameAnimationTimer.java  ForcePanelController.java  ObjectPanelController.java  StatisticsPanelController.java  SurfacePanelController.java | Use-case diagram  \*.fxml  Surface.java  HorizontalVector.java  Css and Images Provider | Detail object diagram  Cube.java  Cylinder.java |

Nhóm chia dự án thành 4 phần riêng biệt để thuận tiện cho việc giao nhiệm vụ cho mỗi thành viên.

* Design:
* General class and controller diagram:
  + Nguyen Trong Nghia 20215101
* Detail class diagram:
  + Model: Le Duy Nghiem 20215102
  + Controller: Nguyen Van Nam 20215097
* Use-case diagram:
  + Pham Thanh Nam 20194630
* Coding:
  + Controller package:
    - Nguyen Trong Nghia 20215101
    - Nguyen Van Nam 20215097
  + Model package:
    - Nguyen Trong Nghia 20215101
    - Pham Thanh Nam 20194630
    - Nguyen Van Nam 20215097
    - Le Duy Nghiem 20215102
  + View package:
    - Pham Thanh Nam 20194630
* Report:
  + Nguyen Trong Nghia 20215101
* Slide:
  + Nguyen Trong Nghia 20215101

# 2 Mini-project description

## 2.1 Project overview

* Trong dự án này, nhóm cần tạo ra một ứng dụng mô phỏng tương tác đơn giản để minh hoạ các định luật chuyển động của Newton.
* Tất cả công việc, nhóm hợp tác và chia sẻ thông tin với nhau bằng cách sử dụng [Github](https://github.com/mrbad12e/OOLT.VN.20231-24.git).
* Để xây dựng ứng dụng này thành công, nhóm cũng xây dựng biểu đồ use-case và các biểu đồ lớp trong quá trình phát triển.
* Các khái niệm lập trình hướng đối tượng như Kế thừa, Đa hình và Liên kết, v.v. cũng được áp dụng vào dự án này.
* Tất cả các ý tưởng cho dự án, cũng như lý do nhóm chọn những ý tưởng đó, cũng cần được giải thích trong báo cáo này cũng như trong bài thuyết trình của nhóm.

## Mini-project requirement

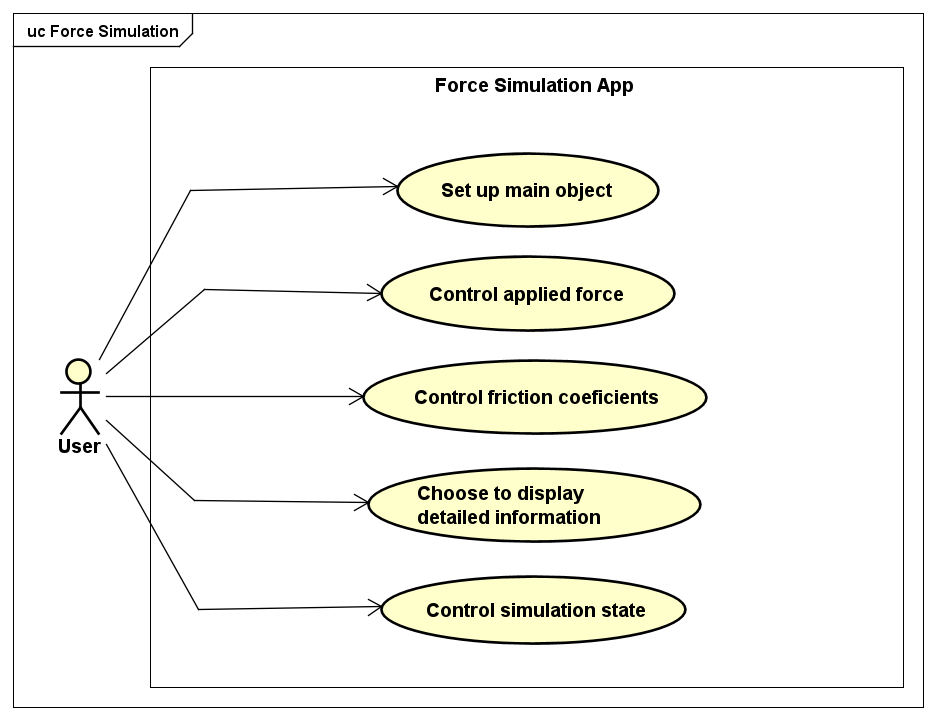
Như đã đề cập trước đó, trong dự án này, nhóm xây dựng một ứng dụng mô phỏng tương tác đơn giản để minh họa các định luật chuyển động của Newton với một số yêu cầu cụ thể:

* Đối với GUI, chương trình nên trông giống như trong tài liệu tham khảo [1], bao gồm hai phần chính, bầu trời và bề mặt.
* Người dùng có thể kiểm soát tất cả các thành phần của hệ thống vật lý, bao gồm một đối tượng chính, bề mặt và một người diễn viên có thể áp dụng một lực ngang lên trung tâm khối của đối tượng và sau đó quan sát chuyển động của đối tượng chính.
* Để bắt đầu ứng dụng, người dùng cần thiết lập đối tượng chính bằng cách kéo một tùy chọn (hoặc hình lập phương hoặc hình trụ) từ menu đối tượng ở góc trái dưới lên bề mặt và sau đó chỉ định các tham số phù hợp dựa trên hình dạng của đối tượng:
  + Cube-shaped object: độ dài cạnh, khối lượng
  + Cylinder-shaped object: bán kính, khối lượng
* Khi mô phỏng đang chạy, người dùng có thể kiểm soát:
  + Actor (được biểu thị bằng lực mà anh ta áp dụng) bằng cách thay đổi chiều dài và hướng của lực ở phần trung tâm dưới cùng bằng cách sử dụng thanh trượt hoặc chỉ định số lực Newton (N) trong ô cửa panel.
  + Các hệ số ma sát tĩnh và ma sát động của bề mặt bằng cách sửa giá trị trong thanh trượt hoặc ô cửa panel ở phần dưới cùng bên phải.
  + Tất cả các số liệu thống kê liên quan, bao gồm giá trị và hướng của tất cả các lực, tổng lực, khối lượng, vận tốc, gia tốc và vị trí của đối tượng chính (bao gồm các đơn vị góc nếu đối tượng có hình trụ) thông qua các ô đánh dấu tương ứng trên bảng điều khiển ở góc phải trên.
  + Tạm dừng, tiếp tục và đặt lại bằng cách mô phỏng.
* Để mô phỏng chuyển động, chương trình tính toán lại số liệu thống kê của đối tượng chính sau mỗi khoảng thời gian Δt và cách lực vật lý ảnh hưởng đến đối tượng dựa trên công thức mà hướng dẫn đã cung cấp.

Hơn nữa, có một số điểm nhóm cũng cần làm rõ trong dự án ngoài một số yêu cầu đã nêu ở trên:

* + Người dùng không thể chọn một đối tượng chính khác vì chúng tôi nghĩ rằng việc người dùng thay đổi đối tượng hình hộp chữ nhật thành đối tượng hình trụ sẽ không có ý nghĩa (thống kê, đơn vị chúng ta cần giữ khi thay đổi hai đối tượng khác nhau là gì?).
  + Người dùng không thể sửa đổi bất kỳ thông số nào của đối tượng hiện tại (nếu chúng tôi có thời gian, chúng tôi sẽ cập nhật chức năng này).
  + Đối với hai chức năng trên, người dùng có thể đặt lại ứng dụng và chọn lại đối tượng chính.

## 2.3 Use case diagram



### 2.3.1 Set up the main object

* Khi thiết lập đối tượng chính: người dùng cần chọn loại đối tượng (hình trụ hoặc hình hộp) bằng cách "kéo và thả", sau đó nhập giá trị cho khối lượng và kích thước của đối tượng.
* Sau khi người dùng nhập tất cả các tham số cho đối tượng, chương trình sẽ tạo, điều chỉnh và hiển thị kích thước của đối tượng đó trên bề mặt.
* Người dùng có thể nhìn thấy sự thay đổi trong kích thước của đối tượng phản ánh theo kích thước cửa sổ.

### 2.3.2 Control the applied force

* Người dùng có thể điều chỉnh thanh trượt hoặc chỉ cần nhập giá trị vào ô văn bản. Nếu người dùng nhập giá trị vào ô văn bản, anh ta phải nhấn Enter để xác nhận giá trị này.
* Sau khi người dùng điều chỉnh lực được áp dụng, chương trình sẽ thay đổi lực được áp dụng vào đối tượng và cập nhật lực net, gia tốc và chiều rộng của vector đại diện cho nó.
* Người dùng có thể thấy rằng đối tượng di chuyển nhanh hơn hoặc chậm hơn, và chiều rộng của vector lực sẽ thay đổi. Các giá trị của nhãn đại diện cho gia tốc và lực sẽ thay đổi tuỳ theo lực được áp dụng hiện tại.

### 2.3.3 Control the friction coefficients

* Tương tự như trường hợp khi người dùng muốn kiểm soát lực được áp dụng, người dùng có thể điều chỉnh thanh trượt hoặc nhập giá trị vào ô văn bản. Nếu người dùng nhập giá trị vào ô văn bản, anh ta phải nhấn Enter để xác nhận giá trị này.
* Sau khi người dùng điều chỉnh các hệ số ma sát, chương trình sẽ cập nhật các thống kê như thay đổi lực ma sát và cập nhật chiều rộng của vector đại diện cho lực ma sát đó.
* Người dùng có thể thấy rằng đối tượng di chuyển nhanh hơn hoặc chậm hơn, và giá trị của các nhãn đại diện cho gia tốc và lực sẽ thay đổi tuỷ theo hệ số ma sát hiện tại.

### 2.3.4 Choose to display detailed information

* Người dùng có thể chọn hiển thị thông tin chi tiết về lực và đối tượng bằng cách đánh dấu vào các ô checkbox.
* Nếu người dùng đánh dấu vào các ô checkbox, chương trình sẽ hiển thị giá trị của các thuộc tính nhãn tương ứng.
* Người dùng có thể quan sát giá trị của những nhãn này nếu anh ta đánh dấu vào các ô checkbox. Nếu anh ta muốn không xem thông tin chi tiết, anh ta có thể bỏ chọn các ô checkbox. Khi quan sát các thống kê, người dùng có thể nắm bắt được hệ thống vật lý.

## 2.3.5 Control simulation state

* Người dùng có thể thay đổi trạng thái mô phỏng như bắt đầu, tạm dừng và tiếp tục bằng cách nhấn các nút ">", "||" và khởi động lại dự án bằng nút "Reset".
* Khi người dùng nhấn các nút, chương trình sẽ thực hiện tương ứng với ý nghĩa của các nút bằng cách tạm dừng/tiếp tục sự chuyển động và quay của các nút hoặc làm mới bảng điều khiển.
* Người dùng có thể thao tác trên bảng điều khiển của chương trình và có thời gian để nghĩ về hiện tượng vật lý bằng cách chọn tạm dừng hoặc tạo hệ thống vật lý mới.

# 3 Design

## 3.1 General class diagram

Trong dự án này, nhóm áp dụng kiến trúc MVC vì lợi ích của nó và để phân công công việc thành viên.

Diagram

Description automatically generated

## 3.2 Package details

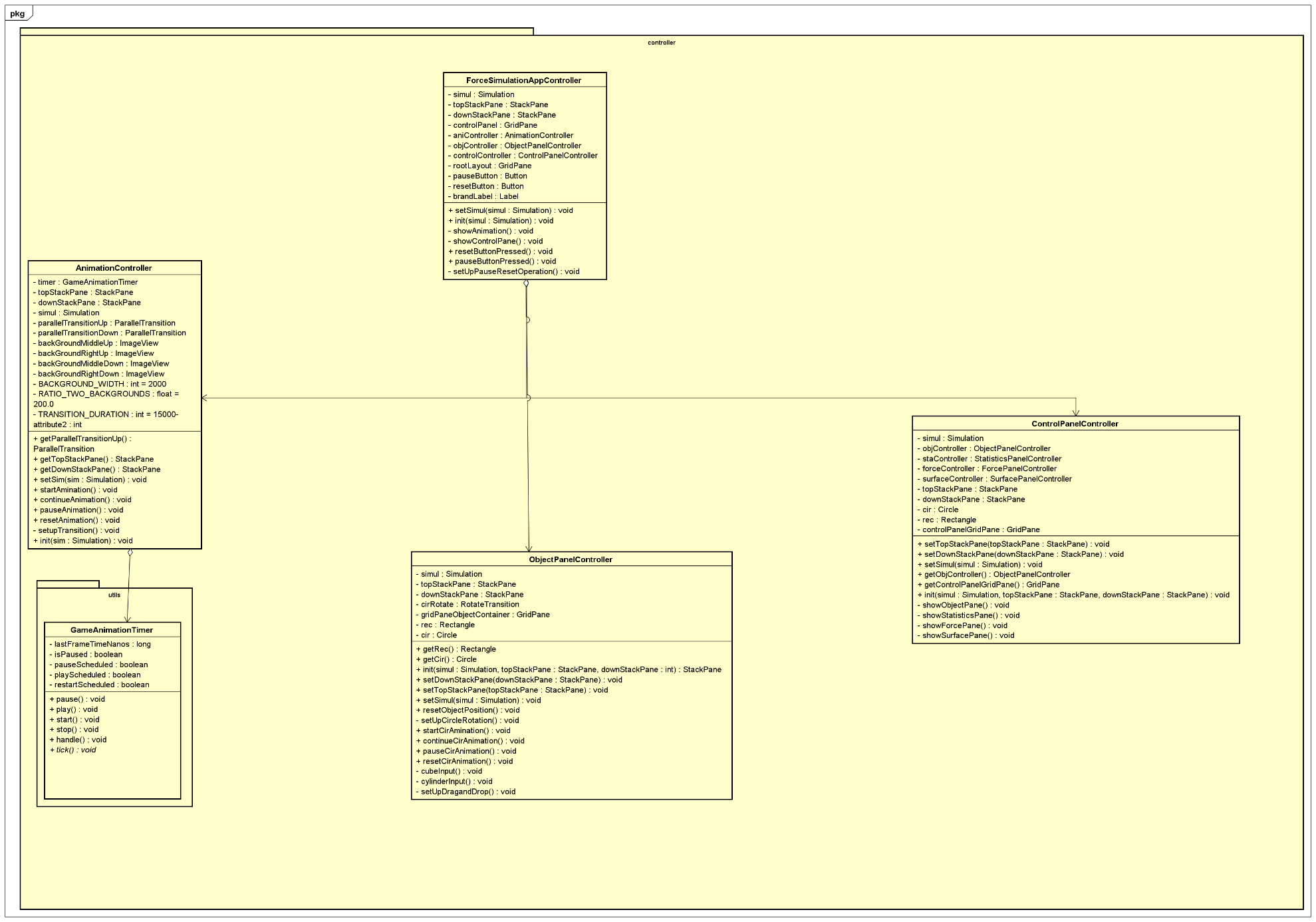
### 3.2.1 Controller package

General class diagram cho gói controller:

Diagram

Description automatically generated

Đầu tiên, chúng tôi sẽ chia bộ điều khiển thành các thành phần nhỏ hơn để thuận tiện và dễ bảo trì. Lớp ForceSimulationAppController là bộ điều khiển chính của dự án, vì vậy chúng tôi có thể chia các nhiệm vụ của nó thành các bộ điều khiển khác như AnimationController, ObjectPanelController và ControlPaneController để quản lý.



* ForceSimulationAppController: Lớp này là bộ điều khiển của tệp 'RootLayout.fxml' và được sử dụng để quản lý màn hình chính, kiểm soát nút đặt lại và tạm dừng mô phỏng. Nó cũng được sử dụng để kết hợp các cảnh vào stage trong chương trình.
  + Thuộc tính:
    - simul: Simulation từ model package.
    - topStackPane: thuộc tính này được sử dụng để chứa 2 hình ảnh đại diện cho góc nhìn phía trên của mô phỏng.
    - downStackPane: thuộc tính này được sử dụng để chứa 2 hình ảnh đại diện cho góc nhìn phía dưới của mô phỏngi, và cũng được sử dụng để kiểm soát kích thước của đối tượng.
    - controlPane: GridPane để chứa bộ điều khiển cho lực ma sát và lực được áp dụng. Nó cũng được sử dụng để chứa hình đối tượng đại diện cho 2 đối tượng (hình hộp chữ nhật và hình trụ).
    - aniController: thực thể của lớp AnimationController.
    - objController: thực thể của lớp ObjectPanelController.
    - controlController: thực thể của lớp ControlPanelController.
    - rootLayout: thực thể của GridPane và được sử dụng như một khung cho hoạt ảnh.
    - pauseButton: thực thể của Button - được sử dụng để tạm dừng hoặc bắt đầu mô phỏng.
    - resetButton: thực thể của Button - được sử dụng để đặt lại mô phỏng.
    - brandLabel: hiển thị tên nhóm.
  + Phương thức:
    - setSimul(Simulation simul): truyền đối tượng mô phỏng vào chương trình mô phỏng.
    - init(Simulation simul): truyền đối tượng mô phỏng vào chương trình và cũng gọi 3 phương thức showAnimation(), showControlPane() và setUpPauseResetOperation() để khởi tạo ba bộ điều khiển này.
    - showAnimation(): thiết lập aniController của lớp AnimationController.
    - showControlPane(): thiết lập thuộc tính controlController của chương trình mô phỏng.
    - setUpPauseResetButton(): thiết lập nút tạm dừng và đặt lại trong topStackPane và các hành vi của nút bấm khi chương trình đang chạy.
    - resetButtonPressed(): đặt lại chương trình mô phỏng.
    - pauseButtonPressed(): tạm dừng mô phỏng nếu nó đã bắt đầu hoặc nó sẽ tiếp tục hoặc bắt đầu chương trình mô phỏng.
* AnimationController: Bộ điều khiển này được sử dụng để kiểm soát tốc độ di chuyển của topStackPane và downStackPane trong mô hình.
  + Thuộc tính:
    - BACKGROUND\_WIDTH = 200: đây là giá trị mặc định cho cả 4 hình ảnh.
    - ratioTwoBackGround = 200.0f
    - TRANSITION\_DURATION = 15000
    - timer: thực thể của lớp GameAnimationTimer - được sử dụng để xử lý mỗi khung hình trong khi mô phỏng đang chạy.
    - simul: thực thể của lớp Simulation.
    - parallelTransitionUp: đưa hình ảnh bầu trời di chuyển song song với mặt đất.
    - parallelTransitionDown: đưa hình ảnh mặt đất di chuyển song song với bầu trời.
    - topStackPane: chứa 2 hình ảnh (đại diện bởi backGroundMiddleUp và backGroundRightUp) - được thiết lập để di chuyển song song trong suốt quá trình mô phỏng. Nó cũng được sử dụng để chứa nhãn và bảng điều khiển ô kiểm cho chương trình.
    - downStackPane: chứa 2 hình ảnh (đại diện bởi backGroundMiddleDown và backGroundRightDown) - được thiết lập để di chuyển song song trong suốt quá trình mô phỏng. Nó cũng được sử dụng để chứa 2 đối tượng (Cube và Cylinder) và Sliders cũng như các thuộc tính TextField để kiểm soát lực được áp dụng cũng như các hệ số ma sát.
    - backGroundMiddleUp: hình ảnh của bầu trời trong mô phỏng.
    - backGroundRightUp: hình ảnh của bầu trời trong mô phỏng.
    - backGroundMiddleDown: hình ảnh của mặt đất trong mô phỏng.
    - backGroundRightDown: hình ảnh của mặt đất trong mô phỏng.
  + Phương thức:
    - getParallelTransitionUp(): lấy parallelTransitionUp - kiểm soát chuyển động của bầu trời.
    - getTopStackPane(): lấy topStackPane của mô phỏng.
    - getDownStackPane(): lấy downStackPane của mô phỏng.
    - init(Simulation sim): khởi tạo con trỏ của thuộc tính simul thành simul của chương trình - thông qua phương thức setSim(Simulation simul) và cũng tạo timer cho mô phỏng.
    - setSim(Simulation simul): khởi tạo con trỏ.
    - startAnimation(): bắt đầu chương trình bằng cách khởi động timer và gọi phương thức play() của parallelTransitionUp và parallelTransitionDown.
    - continueAnimation(): tiếp tục chương trình bằng cách khởi động timer và gọi phương thức play() của parallelTransitionUp và parallelTransitionDown.
    - pauseAnimation(): tạm dừng chương trình bằng cách tạm dừng timer và gọi phương thức play() của parallelTransitionUp và parallelTransitionDown.
    - resetAnimation(): đặt lại chương trình bằng cách dừng tất cả các timer, parallelTransitionUp và parallelTransitionDown của mô phỏng và đặt vị trí của parallelTransitionUp và parallelTransitionDown về điểm bắt đầu.
    - setupTransition(): thiết lập chuyển động của hình ảnh cho chuyển động trong chương trình.
* GameAnimationTimer: Lớp này là một tiện ích cho model - sử dụng cho hoạt ảnh.
  + Thuộc tính:
    - lastFrameTimeNanos: lưu trữ thời gian frame cuối cùng.
    - isPaused: mô phỏng đã tạm dừng.
    - pauseScheduled: mô phỏng được tạm dừng.
    - playScheduled: mô phỏng được phát tiếp.
    - restartScheduled: mô phỏng được khởi động lại.
  + Phương thức:
    - pause(): tạm dừng mô phỏng.
    - play(): chạy mô phỏng.
    - start(): bắt đầu chạy mô phỏng.
    - stop(): dừng mô phỏng.
    - handle(): vì lớp này mở rộng từ lớp AnimationTimer, nó cần triển khai phương thức handle - abstract class trong lớp AnimationTimer.
    - tick(float secondsSinceLastFrame): đây là một abstract class, cập nhật gia tốc, vận tốc và vị trí dựa trên thời gian từ khung hình trước.

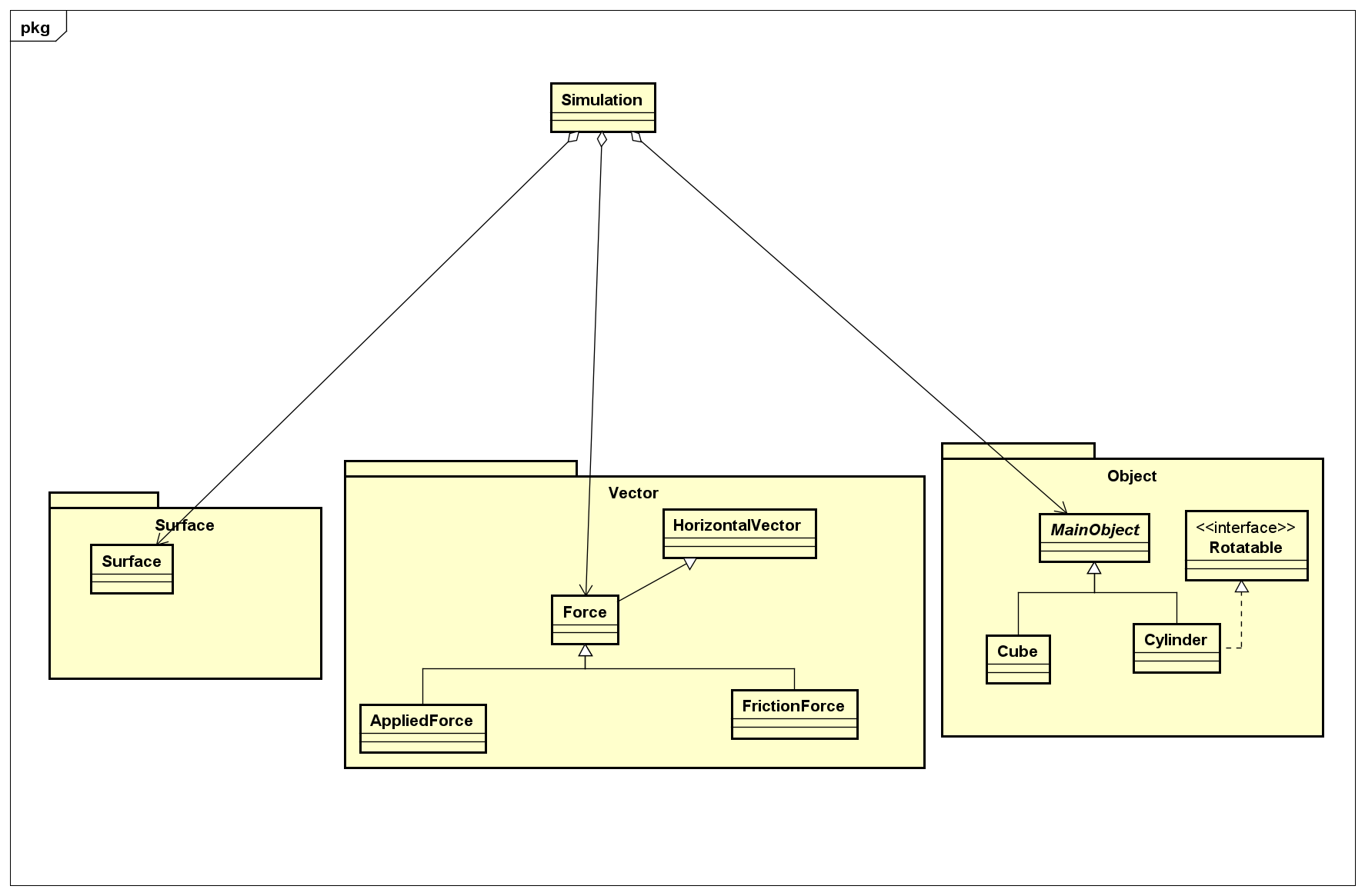
Nhóm chia nhỏ nhiệm vụ của ControlPanelController thành các lớp điều khiển nhỏ hơn vì cần phải kiểm soát lực, bề mặt, đối tượng và cũng biểu diễn thống kê -> tạo ra 4 bộ điều khiển nhỏ độc lập. Dưới đây là controller class diagram. Vì StatisticalPanelController có nhiều thuộc tính có cùng vai trò nên nhóm sẽ ẩn trong sơ đồ lớp và liệt kê chúng khi giải thích chi tiết lớp này.

A picture containing diagram

Description automatically generated

* ControlPanelController: kiểm soát một số giá trị cho mô phỏng như lực, hệ số ma sát, lực, thuộc tính đối tượng, v.v...
  + Thuộc tính:
    - simul: thực thể của lớp Simulation.
    - objController: thực thể của lớp ObjectPanelController- kiểm soát đối tượng trong chương trình.
    - staController: thực thể của lớp StatisticsPanelController - kiểm soát các thống kê trong chương trình.
    - forceController: thực thể của lớp ForcePanelController - kiểm soát lực được áp dụng trong chương trình.
    - surfaceController: thực thể của lớp SurfaceController - kiểm soát hệ số ma sát của bề mặt trong chương trình.
    - topStackPane: thực thể của lớp StackPane - chứa 2 hình ảnh đại diện cho bầu trời trong mô phỏng.
    - downStackPane: thực thể của lớp StackPane - chứa 2 hình ảnh đại diện cho mặt đất trong mô phỏng.
    - cir: đại diện cho hình trụ trong mô phỏng nếu đối tượng chính là hình trụ.
    - rec: đại diện cho hình hộp trong mô phỏng nếu đối tượng chính là hình hộp.
    - controlPanelGridPane: kiểm soát controlPanelGridPane - được sử dụng để chứa ObjectPanel, ForcePanel và SurfacePanel.
  + Phương thức:
    - init(Simulation simul, StackPane topStackPane, StackPane downStackPane): thiết lập các thuộc tính simul, topStackPane và downStackPane của mô phỏng và gọi 4 phương thức showObjectPanel(), showForcePanel(), showSurfacePanel() và showStatisticPanel().
    - setTopStackPane(StackPane topStackPane): nhận topStackPane làm tham số và thiết lập thuộc tính topStackPane của nó thành topStackPane đó.
    - setDownStackPane(StackPane downStackPane): nhận downStackPane làm tham số và thiết lập thuộc tính downStackPane của nó thành downStackPane đó.
    - setSimul(Simulation simul): nhận thực thể của Simulation làm tham số và sau đó gán giá trị của tham số này cho 'simul'.
    - getObjController(): lấy thuộc tính objController của mô phỏng.
    - getControlPanelGridPane(): lấy controlPanelGridPane của mô phỏng.
    - showObjectPane():thiết lập objectPanelController và cũng là bảng điều khiển đối tượng của mô phỏng.
    - showStatisticsPane(): thiết lập statisticPanelController, bảng điều khiển thống kê của mô phỏng.
    - showForcePane(): thiết lập forcePanelController, bảng điều khiển lực của mô phỏng.
    - showSurfacePane(): thiết lập surfacePanelController, bảng điều khiển bề mặt của mô phỏng.
* ObjectPanelController: kiểm soát đầu vào cho khối lượng của đối tượng, bán kính (nếu là hình trụ) hoặc chiều dài cạnh (nếu là hình hộp). Lớp này cũng được sử dụng để kiểm soát xoay nếu đối tượng chính là có thể quay (hình trụ).
  + Thuôc tính:
    - simul: thực thể của lớp Simulation.
    - topStackPane: thực thể của lớp StackPane - chứa 2 hình ảnh đại diện cho bầu trời.
    - downStackPane: thực thể của lớp StackPane - chứa 2 hình ảnh đại diện cho mặt đất.
    - cirRotate: thực thể của RotateTransition, được sử dụng cho hình trụ để kiểm soát quay.
    - girdPaneObjectContainer: đại diện cho GridPane có kích thước (2x1) để chứa 2 đối tượng (hình trụ và hình hộp).
    - rec: đại diện cho đối tượng Cube trong mô phỏng.
    - cir: đại diện cho đối tượng Cylinder trong mô phỏng.
  + Phương thức:
    - getRec(): lấy hình chữ nhật - đại diện cho hình hộp trong mô phỏng.
    - getCir(): lấy hình tròn - đại diện cho hình trụ trong mô phỏng.
    - init(): thiết lập hình ảnh cho hình trụ và hình hộp trong mô phỏng và thiết lập con trỏ của simul bằng cách gọi setSimul(Simulation simul), topStackPane và downStackPane với simul, topStackPane và downStackPane của mô hình Simulation. Sau đó, gọi 2 phương thức setUpDragAndDrop() và setUpCircleRotation().
    - setSimul(Simulation simul): thiết lập con trỏ của thuộc tính simul trong lớp này với simul trong chương trình.
    - setDownStackPane(StackPane downStackPane): thiết lập con trỏ của thuộc tính downStackPane với downStackPane trong mô hình và sẽ liên kết Radius Property của Cylinder, Height và Width Properties của Cube với 30% chiều cao của downStackPane.
    - setTopStackPane(): thiết lập con trỏ của.
    - resetObjectPosition(): đặt lại tất cả các đối tượng trong mô phỏng, đặt lại kích thước và khối lượng của đối tượng về giá trị mặc định cũng như di chuyển nó đến gridPaneObjectContainer.
    - setUpCircleRotation(): thiết lập tham số cho Quay của Circle, sau đó nếu đối tượng của mô hình mô phỏng là Cylinder, liên kết các giá trị góc của hình trụ này với tỷ lệ Rotation.
    - startCirAnimation(): bắt đầu quay của hình trụ trong mô hình mô phỏng.
    - continueCirAnimation(): tiếp tục quay của hình trụ trong mô hình mô phỏng.
    - pauseCirAnimation(): tạm dừng quay của hình trụ trong mô hình mô phỏng.
    - resetCirAnimation(): đặt lại quay của hình trụ trong mô hình mô phỏng bằng cách đặt lại vị trí về điểm bắt đầu và cũng dừng quay.
    - cubeInput(): tạo một hộp thoại để người dùng nhập khối lượng và chiều dài cạnh của hình hộp, sau đó sẽ cập nhật kích thước và khối lượng của đối tượng dựa trên giá trị đã nhập.
    - cylinderInput(): tạo một hộp thoại để người dùng nhập khối lượng và bán kính của hình trụ, sau đó sẽ cập nhật bán kính và khối lượng của đối tượng dựa trên giá trị đã nhập.
    - setUpDragAndDrop(): thiết lập các hành động cho kéo và thả đối tượng. Nếu mô phỏng đang chạy, chúng ta không cho phép kéo đối tượng.
  + Private class:
    - EventDragDetected: Lớp này được sử dụng để bắt sự kiện khi nhấn chuột vào đối tượng. Nó sẽ tạo một lớp phủ trên đối tượng để kéo nó..
* StatisticsPanelController: biểu diễn các thuộc tính của mô hình như lực được áp dụng, lực ma sát, các thuộc tính của đối tượng cũng như vận tốc và gia tốc của nó.
  + Thuộc tính:
    - simul: thực thể của lớp Simulation.
    - stackPane: chứa các ô checkbox và label đại diện cho các thống kê tương ứng.
    - rec: đại diện cho hình trụ trong mô phỏng.
    - cir: đại diện cho hình hộp trong mô phỏng.
    - aArrow: hình ảnh vectơ đại diện cho lực được áp dụng.
    - fArrow: hình ảnh vectơ đại diện cho lực ma sát.
    - nArrow: hình ảnh vectơ đại diện cho lực tổng hợp.
    - angLabel: Label hiển thị giá trị của góc hiện tại.
    - angAccLabel:Label hiển thị giá trị của gia tốc góc hiện tại.
    - angVelLabel:Label hiển thị giá trị của vận tốc góc hiện tại.
    - posLabel: Label hiển thị giá trị của vị trí hiện tại.
    - accLabel:Label hiển thị giá trị của gia tốc hiện tại.
    - velLabel: Label hiển thị giá trị của vận tốc hiện tại.
    - aForceLabel: Label hiển thị giá trị của lực được áp dụng hiện tại.
    - fForceLabel: Label hiển thị giá trị của lực ma sát hiện tại.
    - sumForceLabel: Label hiển thị giá trị của lực tương tổng hiện tại.
    - posCheckBox: CheckBox hiển thị của thuộc tính posLabel.
    - accCheckBox: CheckBox hiển thị của thuộc tính accLabel.
    - velCheckBox: CheckBox hiển thị của thuộc tính velLable.
    - forceCheckBox: CheckBox hiển thị của thuộc tính aArrow và fArrow.
    - sumForceCheckBox: CheckBox hiển thị của thuộc tính nArrow.
    - masCheckBox: CheckBox hiển thị của thuộc tính massLabel.
    - massLabel: hiển thị khối lượng của đối tượng.
    - valueCheckBox: hiển thị của các thuộc tính aArrowLabel, fArrowLabel và nArrowLabel.
    - aArrowLabelhiển thị giá trị của lực được áp dụng hiện tại.
    - fArrowLabel: hiển thị giá trị của lực ma sát hiện tại.
    - nArrowLabel: hiển thị giá trị của lực tổng hợp hiện tại.
  + Phương thức:
    - init(): thiết lập giá trị văn bản ban đầu cho một số thuộc tính Label và thiết lập con trỏ của các thuộc tính cir và rec để trỏ đến cir và rec tương ứng, đồng thời cũng gọi setSimul(), setTopStackPane(), setUpAppliedForce(), setUpFrictionForce(), setUpNetForce().
    - setSimul(Simulation simul): thiết lập con trỏ của thuộc tính simul để trỏ đến simul trong mô hình Simulation.
    - setUpNetForce(): tạo và thiết lập điểm bắt đầu của nArrow ở giữa đối tượng.
    - setUpFrictionForce(): tạo và thiết lập điểm bắt đầu của fArrow ở giữa đối tượng.
    - setUpAppliedForce(): tạo và thiết lập điểm bắt đầu của aArrow ở giữa đối tượng.
    - setTopStackPane(StackPane topStackPane): thiết lập con trỏ của thuộc tính stackPane để trỏ đến topStackPane tương ứng trong chương trình Simulation, sau đó thiết lập vị trí cho các nhãn và giá trị thống kê.
* ForcePanelController: Lớp này được sử dụng để kiểm soát lực tác động, hệ số lực ma sát và lực tổng hợp được áp dụng trong mô hình.
  + Thuộc tính:
    - simul: thực thể của lớp Simulation.
    - forcePanel: StackPane chứa các thuộc tính forceSlider và forceTextField.
    - forceSlider: Slider kiểm soát lực được áp dụng lên đối tượng. Có thể điều chỉnh thanh trượt để thay đổi giá trị của lực được áp dụng.
    - forceTextField: TextField kiểm soát lực được áp dụng lên đối tượng. Có thể nhập giá trị nằm trong khoảng hợp lệ vào ô văn bản để thay đổi lực được áp dụng.
  + Phương thức:
    - init(Simulation simul): đặt con trỏ của simul cùng với simul trong Simulation. Sau đó, liên kết forceTextField và forceSlider với nhau.
    - forceTextFieldOnAction(ActionEvent event): gọi mỗi khi nhấn Enter trên ô TextField, sau đó kiểm tra văn bản và khoảng giá trị hợp lệ, sau đó cập nhật lực được áp dụng của mô phỏng dựa trên giá trị đó.
    - netForceListener(): tuân theo sự thay đổi của lực được áp dụng và lực ma sát, cập nhật lực tổng hợp dựa trên những thay đổi này.
* SurfacePanelController: kiểm soát các thuộc tính của bề mặt trong mô hình, như hệ số ma sát tĩnh và hệ số ma sát động.
  + Thuộc tính:
    - simul: thực thể của lớp Simulation.
    - staticCoefTextField: kiểm soát giá trị của hệ số ma sát tĩnh của lực ma sát trên bề mặt.
    - kineticCoefTextField: kiểm soát giá trị của hệ số ma sát động của lực ma sát trên bề mặt.
    - staticCoefSlider: kiểm soát giá trị của hệ số ma sát tĩnh của lực ma sát trên bề mặt.
    - kineticCoefSlider:kiểm soát giá trị của hệ số ma sát động của lực ma sát trên bề mặt.
  + Phương thức:
    - init(Simulation simul): liên kết giá trị của TextField với các thuộc tính Slider tương ứng - điều khiển hệ số ma sát tĩnh và hệ số ma sát động.
    - surfaceListener(): bắt nhịp bất kỳ thay đổi nào trong giá trị của các hệ số ma sát tĩnh và động, sau đó cập nhật giá trị của lực ma sát dựa trên những giá trị này.
    - staticTextFieldOnAction(ActionEvent event): lấy giá trị của hệ số ma sát tĩnh của bề mặt trong chương trình, kiểm tra xem giá trị mà người dùng nhập có trong khoảng hợp lệ hay không.
    - kineticTextFieldOnAction(ActionEvent event): lấy giá trị của hệ số ma sát động của bề mặt trong chương trình, cũng như để kiểm tra xem giá trị mà người dùng nhập có trong khoảng hợp lệ hay không.

### 3.2.2 Model package



* Simulation: mô hình mô phỏng - kiểm soát bề mặt, đối tượng và lực trong mô phỏng bằng logic.
  + Thuộc tính:
    - isStart: boolean, biểu thị xem mô phỏng đã bắt đầu chưa.
    - isPause: boolean, biểu thị xem mô phỏng đã tạm dừng chưa.
    - obj: đối tượng chính trong mô hình (có thể là hình trụ hoặc hình lập phương).
    - sysVel: Vận tốc của đối tượng khi mô phỏng bắt đầu.
    - sysAcc: Gia tốc của đối tượng khi mô phỏng bắt đầu.
    - sysAngAcc: Gia tốc góc của đối tượng khi mô phỏng bắt đầu.
    - sysAngVel: Vận tốc góc của đối tượng khi mô phỏng bắt đầu.
    - netForce: Lực tổng hợp của mô phỏng.
    - surface: biểu thị bề mặt trong mô phỏng.
    - aForce: biểu thị lực được áp dụng trong mô phỏng.
    - fForce: biểu thị lực ma sát trong mô phỏng.
  + Phương thức khỏi tạo:
    - Simulation(): khởi tạo mô hình với bề mặt, lực tổng hợp = 0 (N) và Lực Ma sát = 0 (N).
    - Simulation(MainObject mainObj, Surface surface, AppliedForce aForce): truyền vào phương thức này lực được áp dụng, bề mặt và đối tượng (có thể là hình lập phương hoặc hình trụ).
  + Phương thức:
    - setSysVel(HorizontalVector horizontalVector): đặt vận tốc của đối tượng .
    - setSysAcc(HorizontalVector horizontalVector): đặt gia tốc của đối tượng.
    - sysVelProperty: lấy thuộc tính vector vận tốc của đối tượng.
    - sysAccProperty: lấy thuộc tính vector gia tốc của đối tượng.
    - getSysAngAcc: lấy thuộc tính gia tốc góc của đối tượng nếu đối tượng là hình trụ.
    - setSysAngAcc(double sysAngAcc): đặt giá trị cho gia tốc góc của đối tượng nếu đối tượng là hình trụ.
    - getSysAngVel(): lấy vận tốc góc của đối tượng hình trụ.
    - setSysAngVel(double sysAngVel): đặt giá trị cho vận tốc góc của đối tượng hình trụ.
    - objProperty(): lấy thuộc tính đối tượng.
    - setObject(MainObject obj): đặt đối tượng (có thể là hình trụ hoặc hình hộp) cho mô hình.
    - getObj(): trả về đối tượng trong mô hình.
    - isPauseProperty(): lấy thuộc tính boolean Pause của mô phỏng.
    - getIsPause(): lấy giá trị boolean của thuộc tính isPause của mô phỏng.
    - setIsPause(boolean isPause): đặt giá trị boolean cho thuộc tính Pause của mô hình.
    - isStartProperty(): lấy thuộc tính boolean Start của mô phỏng.
    - getIsStart(): lấy giá trị boolean của thuộc tính isStart của mô phỏng.
    - setIsStart(boolean isStart): đặt giá trị boolean cho thuộc tính Start của mô hình.
    - getSur(): lấy bề mặt của mô hình.
    - getaForce(): lấy lực được áp dụng của mô hình.
    - setaForce(double aForce): đặt giá trị cho lực được áp dụng của mô hình.
    - getfForce(): lấy lực ma sát của mô hình.
    - getNetForce():lấy lực tổng hợp của mô hình.
    - start(): bắt đầu mô phỏng bằng cách đặt isStart thành True và isPause thành False.
    - pause(): tạm dừng mô phỏng bằng cách đặt isStart thành False.
    - conti(): tiếp tục mô phỏng bằng cách đặt isPause thành False.
    - restart(): đặt lại tất cả thuộc tính của mô phỏng về giá trị ban đầu.
    - updateObjAcc(): cập nhật gia tốc của đối tượng trong mô hình.
    - getObjVel(): lấy thuộc tính vector vận tốc của đối tượng.
    - updateNetForce(): cập nhật lực tổng hợp được áp dụng vào đối tượng bằng cách gọi sumTwoForce().
    - applyForceInTime(double t): cập nhật gia tốc, vận tốc và cả vị trí của đối tượng trong quá trình mô phỏng. Nếu đối tượng là một thể hiện của Cylinder, cập nhật gia tốc góc, vận tốc góc và góc của đối tượng.

#### 3.2.2.1 Object

* MainObject: abstract class, là lớp cơ sở cho 2 đối tượng lớp hình trụ và lớp hình lập phương. Nó chứa các thuộc tính chung cũng như phương thức cho đối tượng trong mô phỏng như khối lượng, vận tốc, gia tốc.
  + Phương thức khỏi tạo:
    - MainObject(): đối tượng sẽ có khối lượng là 50kg mặc định.
    - MainObject(double mass): đặt giá trị khối lượng của đối tượng thành giá trị mass này.
  + Thuộc tính:
    - mass: double - có giá trị mặc định là 50 kg.
    - acc: đại diện cho gia tốc, HorizontalVector và có giá trị mặc định là 0 m/s.
    - vel: đại diện cho vận tốc, HorizontalVector và có giá trị mặc định là 0 m/s².
    - pos: đại diện cho vị trí của đối tượng so với điểm bắt đầu sau khi mô phỏng được bắt đầu.
  + Phương thức:
    - massProperty(): lấy khối lượng của đối tượng dưới dạng thuộc tính.
    - getMass(): lấy giá trị khối lượng của đối tượng.
    - setMass(double mass): đặt giá trị khối lượng cho đối tượng, kiểm tra nếu khối lượng không lớn hơn 0 thì sẽ không chấp nhận giá trị này, và sau đó trả về một Exception.
    - accProperty(): lấy thuộc tính vector gia tốc của đối tượng.
    - velProperty(): lấy thuộc tính vector vận tốc của đối tượng.
    - setAcc(double acc):đặt giá trị cho thuộc tính vector gia tốc của đối tượng.
    - setVel(double vel): đặt giá trị cho thuộc tính vector vận tốc của đối tượng.
    - updateAcc(Force force): cập nhật gia tốc của đối tượng dựa trên lực và khối lượng của đối tượng.
    - updateVel(double t): cập nhật vận tốc của đối tượng dựa trên vận tốc hiện tại và gia tốc, kiểm tra nếu vận tốc giảm và không dương thì đặt nó thành 0.
    - applyForceInTime(Force netForce, Force force, double t): cập nhật gia tốc, vận tốc và cũng vị trí của đối tượng dựa trên lực được áp dụng, thời gian và cũng vận tốc hiện tại của nó.
    - posProperty(): lấy thuộc tính vị trí của đối tượng so với điểm bắt đầu.
    - getPos(): lấy giá trị vị trí của đối tượng so với điểm bắt đầu.
    - setPos(double pos): đặt giá trị cho vị trí của đối tượng.
    - updatePos(double oldVel, double t): cập nhật vị trí của đối tượng dựa trên vận tốc cũ, vị trí và cũng gia tốc cũng như thời gian.
* Rotatable: Đây là một giao diện được sử dụng cho Hình trụ để biểu diễn việc xoay của hình trụ.
  + Phương thức:
    - angAccProperty(): lấy giá trị thuộc tính cho gia tốc góc của đối tượng hình trụ.
    - getAngAcc(): lấy giá trị gia tốc góc của đối tượng hình trụ.
    - setAngAcc(double angAcc): đặt giá trị gia tốc góc cho đối tượng hình trụ.
    - updateAngAcc(Force force): cập nhật gia tốc góc của đối tượng hình trụ dựa trên lực ma sát. Nếu loại lực mà nhập vào không phải là lực ma sát, trả về một Exception.
    - angVelProperty(): lấy giá trị thuộc tính cho vận tốc góc của đối tượng hình trụ.
    - getAngVel(): lấy giá trị vận tốc góc của đối tượng hình trụ.
    - setAngVel(double angle): đặt giá trị cho thuộc tính vận tốc góc của đối tượng hình trụ.
    - updateAngVel(double t): cập nhật vận tốc góc của đối tượng hình trụ.
    - angleProperty(): lấy thuộc tính góc của đối tượng hình trụ.
    - getAngle(): lấy giá trị góc hiện tại của đối tượng hình trụ.
    - setAngle(double angle): đặt giá trị cho góc của đối tượng hình trụ.
    - updateAngle(double oldAngVel, double t): đặt giá trị cho góc của đối tượng hình trụ dựa trên vận tốc góc cũ và thời gian đã trôi qua kể từ khi mô phỏng bắt đầu.
    - radiusProperty(): lấy giá trị thuộc tính của bán kính của hình trụ.
    - getRadius(): lấy giá trị bán kính của hình trụ.
    - setRadius(double radius): đặt giá trị cho bán kính của hình trụ. Khi đặt giá trị cho bán kính, kiểm tra giá trị có nằm trong khoảng hợp lệ hay không. Nếu giá trị này không hợp lệ, trả về một Exception.
    - applyForceInTimeRotate(Force force, double t): sử dụng tương tự như phương thức applyForceInTime, ngoại trừ việc nó cập nhật giá trị cho các thuộc tính góc.
* Cube: kế thừa từ lớp MainObject và thêm một số thuộc tính, phương thức từ lớp MainObject để biểu diễn đối tượng hình hộp trong mô hình mô phỏng.
  + Phương thức khởi tạo:
    - Cube(): kế thừa từ lớp cơ sở MainObject, sau đó nó sẽ đặt giá trị cho chiều dài kích thước mặc định của đối tượng.
    - Cube(double mass): kế thừa từ lớp cơ sở MainObject, sau đó nó sẽ đặt giá trị cho chiều dài kích thước mặc định của đối tượng.
    - Cube(double mass, double size): đặt giá trị cho khối lượng và kích thước của đối tượng.
  + Thuộc tính:
    - size: chiều dài cạnh của hình lập phương và có giá trị mặc định là 0.3.
    - MAX\_SIZE = 1.0: Giá trị tối đa trong khoảng giá trị hợp lệ cho chiều dài cạnh của hình hôp.
    - MIN\_SIZE = 0.1: Giá trị tối thiểu trong khoảng giá trị hợp lệ cho chiều dài cạnh của hình hộp.
  + Phương thức:
    - sizeProperty(): lấy giá trị thuộc tính của kích thước của đối tượng.
    - getSize(): lấy giá trị của kích thước của đối tượng.
    - setSize(double size): đặt giá trị cho kích thước của đối tượng.
* Cylinder: kế thừa từ lớp MainObject và thêm một số thuộc tính, phương thức từ lớp MainObject để biểu diễn đối tượng hình trụ trong mô hình mô phỏng.
  + Phương thức khởi tạo:
    - Cylinder(): kế thừa từ lớp cơ sở MainObject và sau đó sẽ đặt giá trị cho bán kính với giá trị mặc định.
    - Cylinder(double mass): kế thừa từ lớp cơ sở MainObject và đặt giá trị cho bán kính với giá trị mặc định.
    - Cylinder(double mass, double radius): đặt giá trị cho khối lượng và bán kính của đối tượng.
  + Thuộc tính:
    - angle: góc của đối tượng dưới dạng double.
    - angAcc: gia tốc góc của đối tượng dưới dạng double.
    - angVel: vận tốc của đối tượng dưới dạng double.
    - radius: bán kính của đối tượng dưới dạng double và có giá trị mặc định là 0.3.
    - MAX\_RADIUS = 1.0: Giá trị tối đa trong khoảng giá trị hợp lệ cho bán kính.
    - MIN\_RADIUS = 0.1: Giá trị tối thiểu trong khoảng giá trị hợp lệ cho bán kính.
  + Phương thức:
    - angAccProperty(): lấy giá trị thuộc tính kiểu double của gia tốc góc của hình trụ.
    - getAngAcc(): lấy giá trị của gia tốc góc của hình trụ.
    - setAngAcc(double angAcc): đặt giá trị cho gia tốc góc của hình trụ.
    - updateAngAcc(Force force): cập nhật giá trị gia tốc góc dựa trên lực ma sát của nó.
    - angVelProperty(): lấy giá trị kiểu double của vận tốc góc của hình trụ.
    - getAngVel(): lấy giá trị của vận tốc góc của hình trụ.
    - setAngVel(double angVel): đặt giá trị cho vận tốc góc của hình trụ.
    - updateAngVel(double t): cập nhật giá trị vận tốc góc của đối tượng hình trụ dựa trên thời gian đã trôi qua.
    - angleProperty(): lấy giá trị thuộc tính góc của đối tượng hình trụ.
    - getAngle(): lấy giá trị góc của đối tượng hình trụ.
    - setAngle(double angle): đặt giá trị cho góc của đối tượng hình trụ.
    - updateAngle(double oldAngle, double t): cập nhật giá trị góc của đối tượng hình trụ dựa trên góc cũ và thời gian đã trôi qua.
    - radiusProperty(): lấy giá trị thuộc tính bán kính của hình trụ.
    - getRadius(): lấy giá trị bán kính của hình trụ.
    - setRadius(double radius): đặt giá trị cho bán kính của hình trụ, kiểm tra tính hợp lệ của giá trị được nhập vào.
    - applyForceInTimeRotate(Force force, double t): cập nhật gia tốc góc, vận tốc góc và cũng góc dựa trên lực hiện tại và thời gian đã trôi qua.
    - applyForceInTime(Force force, double t): gọi hai phương thức applyForceInTime() từ lớp MainObject và applyForceInTimeRotate() để cập nhật giá trị cho vị trí, vận tốc và gia tốc của đối tượng Hình trụ.

#### 3.2.2.2 Surface

* Surface: đại diện cho bề mặt trong mô hình - được sử dụng cho logic của các thuộc tính của bề mặt.
  + Phương thức khởi tạo:
    - Surface(): khởi tạo một đối tượng bề mặt với các giá trị mặc định.
    - Surface(double staCoef): khởi tạo một đối tượng bề mặt mới và đặt hệ số tĩnh staCoef, trong khi hệ số động được đặt staCoef/2.
    - Surface(double staCoef, double kiCoef): tạo một đối tượng bề mặt mới và đặt hệ số tĩnh staCoef, và hệ số động kiCoef.
  + Thuộc tính:
    - staCoef: hệ số tĩnh của bề mặt.
    - kiCoef: hệ số động của bề mặt.
    - MAX\_STA\_COEF: giá trị tối đa cho hệ số tĩnh và cũng hệ số động của bề mặt.
    - STEP\_COEF:giá trị tối thiểu cho các hệ số ma sát của bề mặt.
  + Phương thức:
    - staCoefProperty(): lấy giá trị thuộc tính của hệ số tĩnh của bề mặt.
    - getStaCoef(): lấy giá trị của hệ số tĩnh của bề mặt.
    - kiCoefProperty(): lấy giá trị thuộc tính của hệ số động của bề mặt.
    - getKiCoef(): lấy giá trị của hệ số động của bề mặt.
    - setStaCoef(double staCoef): đặt giá trị cho hệ số tĩnh của lực ma sát của bề mặt.
    - setKiCoef(double kiCoef): đặt giá trị cho hệ số động của lực ma sát của bề mặt.

#### 3.2.2.3 Vector

* HorizontalVector: biểu diễn vector
  + Phương thức khởi tạo:
    - HorizontalVector(double value): khởi tạo giá trị vector
  + Thuộc tính:
    - direction: boolean để biểu diễn hướng của vector. Nếu giá trị là true, thì vector này sẽ trỏ theo hướng (+), ngược lại nếu là false thì nó sẽ trỏ theo hướng (-).
    - value: giá trị của vector.
  + Phương thức:
    - directionProperty(): lấy thuộc tính boolean của hướng của vector.
    - getDirection(): lấy giá trị boolean của hướng của vector.
    - setDirection(boolean isRight): đặt giá trị thuộc tính boolean của hướng.
    - valueProperty(): lấy thuộc tính value của vector.
    - getValue(): lấy giá trị của vector.
    - setValue(double value): đặt giá trị cho thuộc tính value của vector.
    - getLength(): lấy giá trị tuyệt đối của vector (bỏ qua dấu).
    - updateValueDirection(): cập nhật hướng của vector thành true nếu nó trỏ vào hướng (+), hoặc false nếu nó trỏ vào hướng (-).
    - updateDirectionValue(): cập nhật giá trị của vector thành (+) nếu nó trỏ vào hướng phải, hoặc (-) nếu nó trỏ vào hướng trái.
* Force: Biểu diễn lực
  + Phương thức khởi tạo:
    - Force(double value): kế thừa từ lớp cơ sở của nó - đặt giá trị cho thuộc tính kiểu double của vector - được kế thừa từ HorizontalVector.
    - sumTwoForce(Force f1, Force f2): static method, kết hợp 2 lực. Nó sẽ kết hợp giá trị của f1 và f2, và sau đó, dựa trên giá trị này, nó sẽ tạo một Lực tổng hợp mới trong mô phỏng.
* FrictionForce: mở rộng từ lớp Force, biểu diễn lực ma sát trong mô hình mô phỏng.
  + Phương thức khởi tạo:
    - FrictionForce(double value): tạo một lực ma sát mới và đặt giá trị của nó thành value.
    - FrictionForce(double value, Surface surface, MainObject mainObj, AppliedForce aForce): đặt Surface, đối tượng và lực được áp dụng thành `surface`, `mainObj` và `aForce` tương ứng. Sau đó, dựa trên loại đối tượng, nó sẽ cập nhật lực ma sát tương ứng.
  + Thuộc tính:
    - surface: biểu diễn bề mặt trong mô hình mô phỏng.
    - mainObj: biểu diễn đối tượng trong mô hình mô phỏng.
    - aForce: biểu diễn lực được áp dụng trong mô hình mô phỏng.
    - g = 10: Giá trị gia tốc của trọng trường (giá trị mặc định).
  + Phương thức:
    - updateFrictionForce(): cập nhật lực ma sát dựa trên loại đối tượng trong mô phỏng.
    - setMainObj(MainObject obj): đặt đối tượng của mô phỏng.
* AppliedForce: mở rộng từ lớp Force, biểu diễn lực được áp dụng trong mô hình mô phỏng.
  + Phương thức khởi tạo:
    - AppliedForce(double value): tạo một lực áp dụng mới và sử dụng constructor từ lớp Force để đặt giá trị cho lực áp dụng.
  + Phương thức:
    - setValue(double value): đặt giá trị cho lực áp dụng. Nếu lực áp dụng không nằm trong khoảng giá trị hợp lệ, thì giá trị của lực áp dụng sẽ được đặt thành giá trị mặc định của nó, tới giá trị tối đa hoặc tối thiểu của nó; ngược lại, nó sẽ đặt giá trị của lực áp dụng thành giá trị được chuyển vào.

### 3.2.3 View

Trong dự án này, vì có nhiều thành phần, nhóm chia nó thành nhiều thành phần độc lập để thuận tiện cho việc bảo trì. Điều này giúp nếu muốn cập nhật một số thứ từ một cảnh, nó sẽ không ảnh hưởng đến các thành phần khác. Thứ tự mà các tệp FXML dưới đây là thứ tự mà chúng tôi thêm các cảnh vào stage.

* RootLayout.fxml: Trong RootLayout, tạo 2 GridPane kích thước (1x2) cho các thành phần topStackPane và downStackPane. Sau đó, thêm một AnchorPane chứa các nút reset và play vào Pane(0,0).
* Animation.fxml: Sau khi thiết lập bố cục cho mô phỏng, thêm hình ảnh vào Scene bằng cách thêm 2 StackPane chứa các hình ảnh đại diện cho bầu trời và mặt đất. Trên topStackPane, thêm 2 hình ảnh tương tự đại diện cho bầu trời và cũng 2 hình ảnh tương tự trên downStackPane đại diện cho mặt đất.
* ControlPane.fxml: tạo một GridPane mới (3x1) cho mục đích điều khiển đối tượng, lực và bề mặt. CThêm vào downStackPane, phía trên 2 hình ảnh vì không cần tương tác với các hình ảnh nữa, chỉ điều khiển tốc độ của chúng khi mô phỏng đang chạy.
* ObjectPanel.fxml: Ở đây, tạo một GridPane kích thước (2x1) để chứa các đối tượng Cube và Cylinder, tương ứng và sau đó thêm GridPane này vào downStackPane tại chỉ mục (0,0).
* StatisticsPanel.fxml: Trong cảnh này, tạo một StackPane và đặt con trỏ của StackPane đó vào topStackPane của RootLayout. Sau đó, thêm các nhãn và các ô đánh dấu - điều khiển biểu diễn của các giá trị của lực, vận tốc, gia tốc, ... trong mô phỏng.
* ForcePanel.fxml: Ở đây, tạo một GridPane kích thước (3x1) - chứa một nhãn Force, một ô văn bản để người dùng nhập giá trị cho lực áp dụng và một thanh trượt để người dùng điều chỉnh lực. Sau đó, thêm GridPane này vào downStackPane tại chỉ mục (1,0).
* SurfacePanel.fxml: Cuối cùng, để kiểm soát hệ số ma sát tĩnh và động của bề mặt, tạo một GridPane kích thước (3x3) - chứa 2 ô văn bản và thanh trượt để điều chỉnh các hệ số ma sát của bề mặt, và thêm GridPane này vào downStackPane tại chỉ mục (2,0).

# 4.References

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

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

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

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

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

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