23/12/2021

Vector

Sommario

[Introduzione 2](#_Toc91157936)

[1.1 Informazioni sul progetto 2](#_Toc91157937)

[1.2 Abstract 2](#_Toc91157938)

[1.3 Scopo 2](#_Toc91157939)

[2 Analisi 3](#_Toc91157940)

[2.1 Analisi del dominio 3](#_Toc91157941)

[2.2 Analisi e specifica dei requisiti 3](#_Toc91157942)

[2.3 Use case 5](#_Toc91157943)

[2.4 Pianificazione 5](#_Toc91157944)

[2.5 Analisi dei mezzi 6](#_Toc91157945)

[2.5.1 Software 6](#_Toc91157946)

[2.5.2 Hardware 6](#_Toc91157947)

[3 Progettazione 6](#_Toc91157948)

[3.1 Design delle interfacce 6](#_Toc91157949)

[3.2 Design procedurale 7](#_Toc91157950)

[4 Implementazione 8](#_Toc91157951)

[4.1 Inizio 8](#_Toc91157952)

[4.2 Classi 8](#_Toc91157953)

[4.2.1 Vector.java 8](#_Toc91157954)

[4.2.2 Body.java 8](#_Toc91157955)

[4.2.3 VectorFrame.java 9](#_Toc91157956)

[5 Test 12](#_Toc91157957)

[5.1 Protocollo di test 12](#_Toc91157958)

[5.2 Risultati test 14](#_Toc91157959)

[5.3 Mancanze/limitazioni conosciute 14](#_Toc91157960)

[6 Consuntivo 15](#_Toc91157961)

[7 Conclusioni 15](#_Toc91157962)

[7.1 Sviluppi futuri 15](#_Toc91157963)

[7.2 Considerazioni personali 15](#_Toc91157964)

[8 Bibliografia 15](#_Toc91157965)

[8.1 Sitografia 15](#_Toc91157966)

[9 Allegati 16](#_Toc91157967)

# Introduzione

## Informazioni sul progetto

Sono Andrea Masciocchi e sono l’allievo che ha portato a termine questo progetto.  
I docenti supervisori sono Geo Petrini e Luca Muggiasca.

Sono studente della SAM Trevano, nella classe I3AA e questo progetto è stato commissionato per la lezione del modulo 306, con data di inizio 9 settembre e data di consegna 23 dicembre.

## Abstract

Questo progetto ha lo scopo di fornire un simulatore di forze vettoriali in 2D con interfaccia grafica, che sia in grado di calcolare il vettore risultante dai vettori immessi e animare il movimento di un corpo soggetto a tali forze.  
L’immissione può avvenire in due modi: inserendo i valori singoli manualmente o tramite input del mouse.

## Scopo

Lo scopo didattico del progetto è stato quello di imparare a gestire un progetto da soli, probabilmente abbiamo lavorato singolarmente per avvicinarci gradualmente a quelli che saranno i prossimi progetti con più persone facenti parte del gruppo.

Abbiamo messo in pratica le conoscenze acquisite l’anno scorso nel modulo 431, dove abbiamo imparato ad usare il software Microsoft Project e a gestire un breve progetto, e ne abbiamo acquisite di nuove per quanto riguarda la pianificazione e gestione completa.

Tra le altre cose c’è anche la creazione di una documentazione, che non avevamo mai fatto prima in modo così dettagliato.

Oltre agli scopi didattici riguardanti la materia troviamo anche una parte di autogestione e autodidattica che probabilmente mancava in altre materie, ma con questo progetto siamo andati ad aggiungere tali elementi all’ambiente scolastico.

Lo scopo del progetto invece non è veramente di grande rilevanza.

Non semplifica di molto l’operazione del calcolo di un vettore risultante, e non è nemmeno utilissimo vedere il corpo soggetto a forze muoversi.

Nonostante ciò, credo comunque che sia stato utile come esperienza personale e in generale era un bel progetto da svolgere, interessante per quanto riguarda la programmazione di esso.

# Analisi

## Analisi del dominio

Il prodotto nasce come progetto commissionato dai docenti responsabili, da consegnare per valutarci.

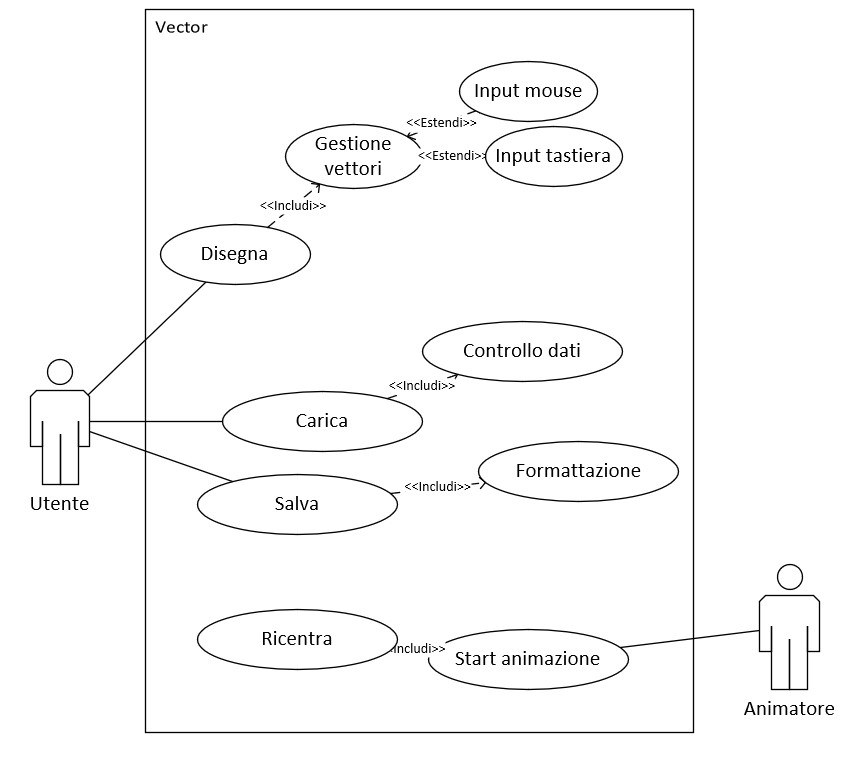
Esistono già prodotti simili su internet, ne ho trovati alcuni un po’antiquati ma comunque funzionanti.

Questo prodotto può essere eseguito su qualsiasi macchina che ha installato java, tramite l’eseguibile java.

## Analisi e specifica dei requisiti

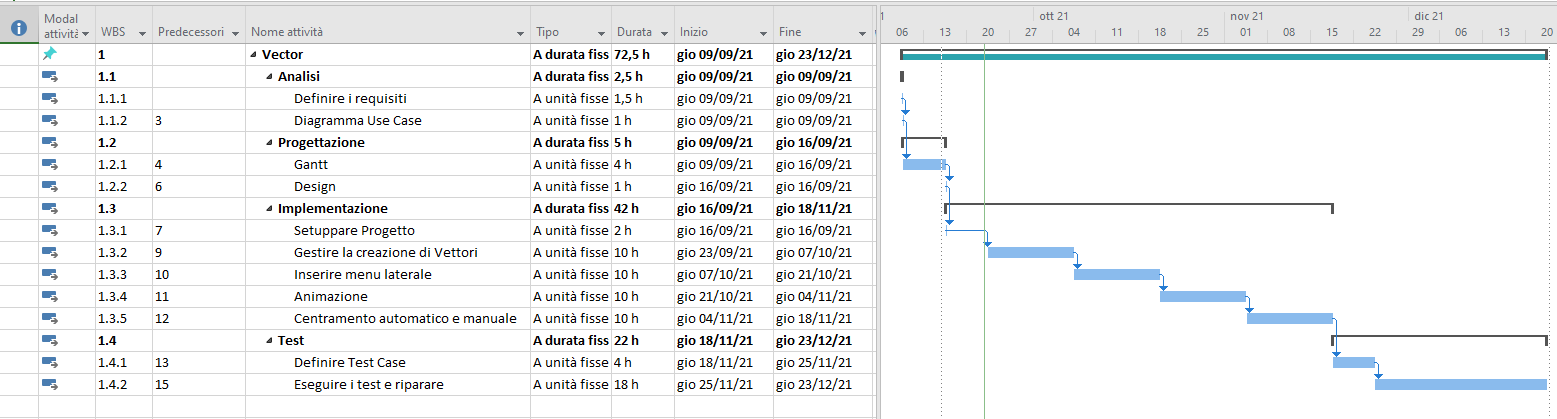
|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Due sezioni all’interno della GUI |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Sezione centrale accetta input mouse per il disegno di vettori |
| **002** | Sezione destra con input per la creazione di vettori dando i valori |
| **003** | La gui è responsive |
| **ID: REQ-02** | |
| **Nome** | Gestione di vettori |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Saranno disegnati nella parte centrale della gui |
| **002** | Sarà possibile nasconderli e farli riapparire |
| **003** | Modificandoli nel menu laterale si modificheranno anche nella gui |
| **004** | Potranno essere modificati nel tempo, mentre l’animazione è in funzione |
| **005** | Potranno essere colorati a scelta dell’utente |
| **ID: REQ-03** | |
| **Nome** | Animazione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Fatta partire l’animazione il corpo si muove secondo il vettore risultante |
| **002** | Il vettore risultante sarà mostrato quando viene fatta partire l’animazione, sarà ben visibile |
| **003** | Il corpo lascia dietro di sé una scia |
| **ID: REQ-04** | |
| **Nome** | Ricentrare la schermata |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Ci sarà un tasto che permette di rimettere al centro il corpo |
| **002** | Se il corpo esce dallo schermo si zoomma all’indietro per far si che rimanga visibile |
| **ID: REQ-05** | |
| **Nome** | Caricare/salvare dati |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | L’utente può salvare/caricare i dati con un apposito tasto |
| **002** | Al caricamento viene controllato il file in modo che non vengano inseriti dati errati |
| **003** | Al salvataggio viene formattato il file in modo che al caricamento sia leggibile |

## Use case



## Pianificazione

In questo progetto ho utilizzato un modello di pianificazione waterfall.



## Analisi dei mezzi

### Software

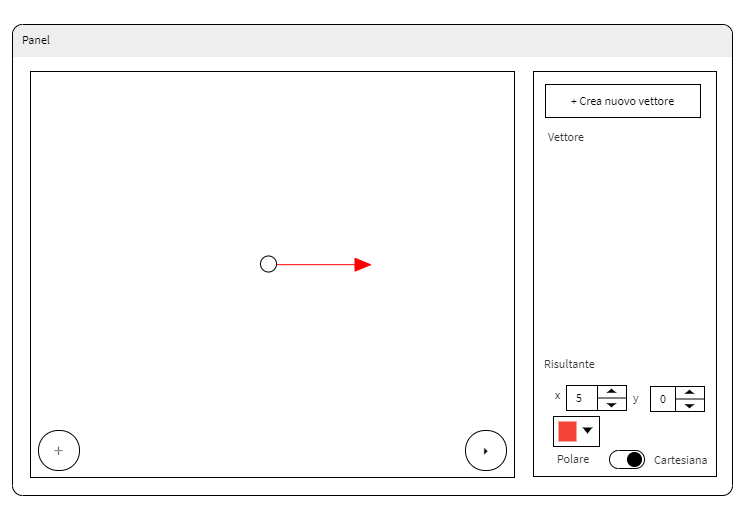
Per questo progetto sono stati utilizzati l’IDE Apache NetBeans 12.5, la versione della JDK è la 16.

### Hardware

L’hardware utilizzato durante lo sviluppo è il PC scolastico e il mio portatile personale.

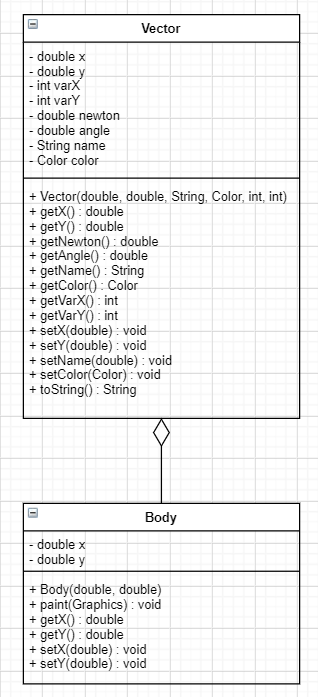
# Progettazione

## Design delle interfacce



## Design procedurale

Diagramma delle classi UML:



# Implementazione

## Inizio

Lavorando con java, e trovandomi molto bene con l’IDE netbeans, utilizzato quest’anno nel modulo 226B, ho optato per questo ambiente di sviluppo.

Perciò inizialmente ho creato il progetto netbeans.

## Classi

Di seguito ho scritto le varie classi

### Vector.java

Vector ha il compito di rappresentare un vettore, i suoi attributi sono:

double x: rappresenta la coordinata x del vettore, in sistema cartesiano

double y: rappresenta la coordinata y del vettore, in sistema cartesiano

int varX: rappresenta la variazione di x che avviene ogni frame dell’animazione

int varY: rappresenta la variazione di x che avviene ogni frame dell’animazione

double newton: rappresenta il modulo del vettore, in sistema polare

double angle: rappresenta l’angolo del vettore, in sistema polare

String name: è il nome del vettore

Color color: è il colore con cui viene rappresentato il vettore

Vector ha un solo costruttore, esso prende due valori in cartesiano e fa il calcolo per convertirle in polare, in tal modo è possibile rappresentarlo in entrambi i sistemi.

Vector contiene inoltre getter e setter per tutti i suoi attributi, e sovrascrive il metodo toString() per poter rappresentare il vettore a terminale per questioni di debugging.

### Body.java

Body è la rappresentazione del corpo soggetto alle forze.

Esso, infatti, contiene la lista di vettori, quindi la lista delle forze al quale il corpo è soggetto.

Oltre alla lista contiene gli attributi double x e double y, che rappresentano le coordinate a cui si trova il corpo.

Il costruttore di body accetta due valori double che sono la lunghezza del canvas, per decidere dove il body andrà posizionato quando istanziato.

Body ha anche un metodo paint() che serve per stampare il corpo, ovvero un pallino rosso, che viene richiamato poi nel VectorFrame.paint().

### VectorFrame.java

VectorFrame è la classe principale di tutto questo programma.

#### Attributi

Gli attributi di questa classe sono parecchi.

**ArrayList<Line2D> lines**: contiene le linee di scia lasciate dal vettore

**double xDrag e double yDrag**: questi contengono le coordinate di dove il mouse è stato draggato/rilasciato, e servono per rappresentare il vettore mentre si dragga e poi per assegnare il valore al vettore quando si rilascia.

**double centerX e double century**: contengono le coordinate del punto centrale del canvas.

**double preBodyX e double preBodyY**: contengono le coordinate del body prima di essere spostato dall’animazione, e servono per disegnare le linee di scia del body

**double canvas e double canvas**: contengono la grandezza del canvas.

**double xOffset e double yOffset**: servono per far funzionare il resize.

**double zoomFactor**: questo contiene il fattore di zoom a cui si è ora, lo cambia l’utente muovendo la rotellina.

**double prevZoomFactor**: contiene lo zoom precedente.

**boolean zoomer**: se sta venendo utilizzata la rotellina questo è true, altrimenti false.

**boolean startDrag**: se sis ta draggando sul canvas questo è true.

**boolean isPolar**: se le coordinate polari sono selezionate questo è true.

**boolean firstTime**: se è il primo paint è true, altrimenti false.

**boolean hiddenVectors**: se si vogliono nascondere i vettori è true, altrimenti false.

**Body body**: contiene un’istanza di body contenuta nel canvas.

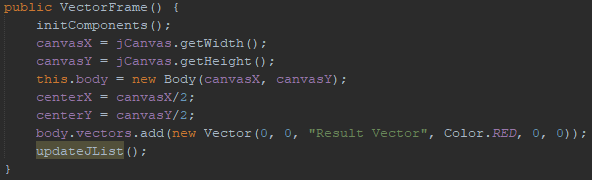
**BufferedImage img**: serve per il sistema di double buffering, renderizza l’immagine prima di proiettarla a schermo

**Graphics2D imgG**: contiene la grafica del JFrame.

**Timer timer e ActionListener act**: per far funzionare il timer che gestisce l’animazione.

**AffineTransform at**: serve per far zoommare.

#### Costruttore



Il costruttore inizializza i componenti di base autogenerate, poi prende la grandezza del canvas per definire dove visualizzare il body.

Da il valore di dove si trova body in centerX e center.

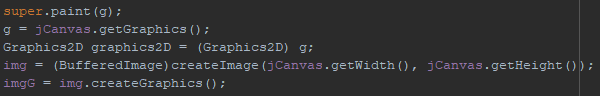
Crea il vettore risultante, che è sempre presente con valori default e aggiorna la lista di vettori in modo da visualizzarlo.

#### Metodi

In questa sezione mostrerò I metodi più importanti.

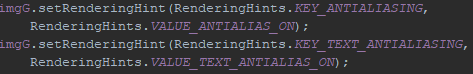
**Paint**

Inizierei descrivendo il metodo paint, che considero parecchio importante essendo che questa classe esporta JFrame ed è costruita per essere l’interfaccia grafica.

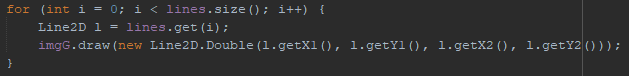


In queste prime 5 righe richiamo il super.paint per ridisegnare ogni volta l’intera interfaccia, poi però cambio il valore di g, al quale assegno la grafica del canvas, dato che tutte le operazioni di disegno si svolgono lì.

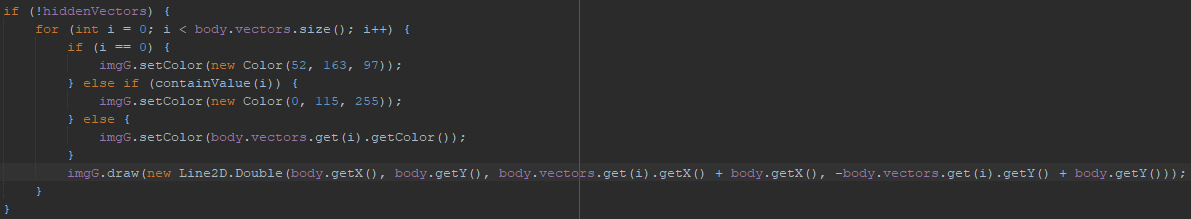
Inoltre inserisco in img un oggetto BufferedImage che servirà per far funzionare il double buffering (che permette all’applicativo di non “sfarfallare”) e in imgG ci creo l’oggetto Graphics corrispondente.



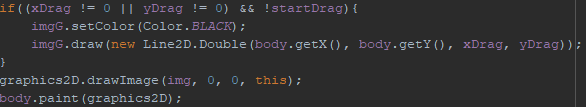
Nell’interfaccia ho anche implementato un sistema di antialiasing gestito dai metodi di Graphics2D, ciò permette di avere delle figure disegnate in modo più lineare e senza mostrare pixel.



Con questo ciclo vado a disegnare ogni linea di scia che ha lasciato il body finora.



In questa pare di codice disegno i vettori a cui è soggetto il corpo (se l’utente vuole visualizzarli) e li coloro in modo diverso in base a se è selezionato o se è il vettore risultante, se non fosse nessuno dei due, controllo che colore gli è stato assegnato dall’utente.



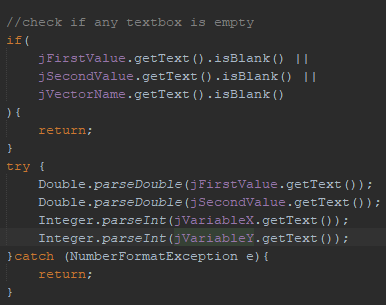
Nel caso l’utente stesse draggando, disegno il vettore nero come anteprima di quello che sta facendo.

Infine disegno l’immagine per terminare l’operazione del double buffering e disegno il body usando il paint presente nella classe body.

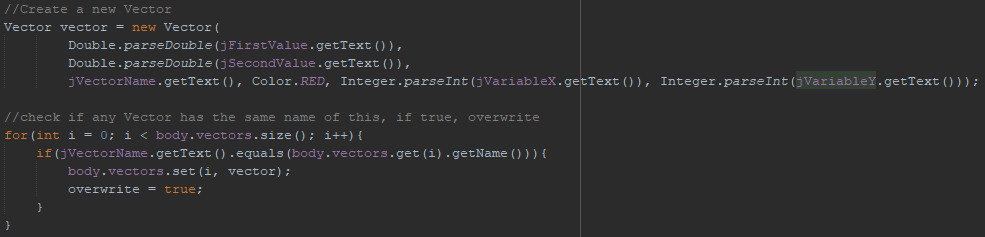
**addVector**

Nel metodo addVector faccio in modo che i valori immessi dall’utente creino il vettore e che venga aggiunto nella lista.

Questo metodo viene richiamato quando si rilascia il mouse nel canvas o quando viene premuto il tasto New Vector.



Nella prima parte del metodo controllo che i valori immessi siano validi, nel caso non lo fossero i valori non cambiano e non viene eseguita nessuna altra parte del metodo.



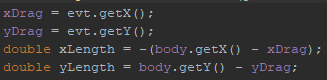
Se i valori sono validi e tutto funziona, procedo nella creazione del vettore con i dati forniti.

Poi passo la lista di vettori e controllo che nessuno di loro abbia lo stesso nome di quello che sta venendo creato, nel caso i nomi corrispondessero il vettore precedentemente presente viene sovrascritto.

A seguire vengono chiamati i metodi per ricalcolare il vettore risultante, per aggiornare la lista di vettori, per resettare le textbox e per ridisegnare nel canvas.

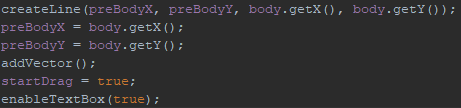
**jCanvasMouseReleased**

In questo metodo si ascolta l’evento mousereleased, che avviene quando l’utente rilascia il tasto del mouse.



Qui assegno a xDrag e yDrag le coordinate di dove il mouse è stato rilasciato, dopodiché calcolo la lunghezza della linea in x e y.

Di seguito assegno i vari valori e li metto nelle textbox. Se il nome è vuoto setta “AutoName” + il numero del vettore.

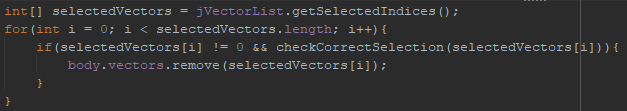


Se il body stesse andando creerei una linea dimodo da non lasciare buchi nella scia, altrimenti la creerebbe comunque ma sarebbe puntiforme e non sarebbe un problema, e do la posizione del body a preBodyX e Y, seguendo la stessa logica di prima non sarebbe un problema se il body fosse fermo.

Richiamo il metodo per aggiungere il vettore alla lista, resetto il valore di startDrag e finisco facendo in modo che l’utente possa scrivere nelle textbox di nuovo.

**jDeleteVectorActionPerformed**:

In questo metodo elimino i vettori selezionati dall’utente nella lista.



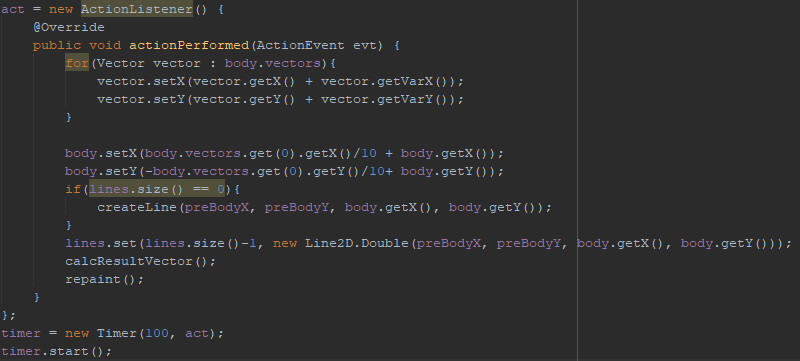
Qui verifico quali vettori sono selezionati dall’utente, metto gli indici di essi nell’array selectedVectors.

Ciclo tutti gli indici e rimuovo il vettori da rimuovere.

In seguito ho fatto in modo di rinominare i vettori con il nome creato automaticamente in modo da non creare buchi con i numeri di questi.

**jStartAnimationActionPerformed**:

In questo metodo viene fatta l’animazione del body che si muove



Nel metodo actionPerformed viene compiuta l’animazione, iniziando dall’aggiunta del valore variabile alle coordinate di ogni vettore.

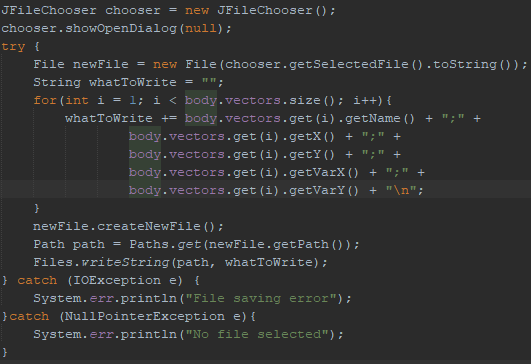
Di seguito setto la X e la Y del body in modo che il movimento sia 10 volte più piccolo della dimensione intera del vettore risultante.

Creo la linea di scia e ricalcolo il vettore risultante nel caso fosse cambiato, infine ridisegno il canvas.

Tutto questo viene eseguito grazie ad un timer che faccio partire quando il tasto start viene premuto, ed esegue l’actionPerformed ogni 100ms.

**jSaveActionPerformed**

Questo metodo, che viene eseguito quando si clicca il tasto “save” serve a salvare la lista dei vettori in un file con tutte le loro informazioni, tra di loro separate da un ‘;’



Creo un’istanza di JFileChooser e richiamo il metodo di istanza showOpenDialog(), e gli passo come argomento ‘null’, in questo modo si aprirà un popup dove potrò selezionare il percorso dove salvare il file e dargli il nome.

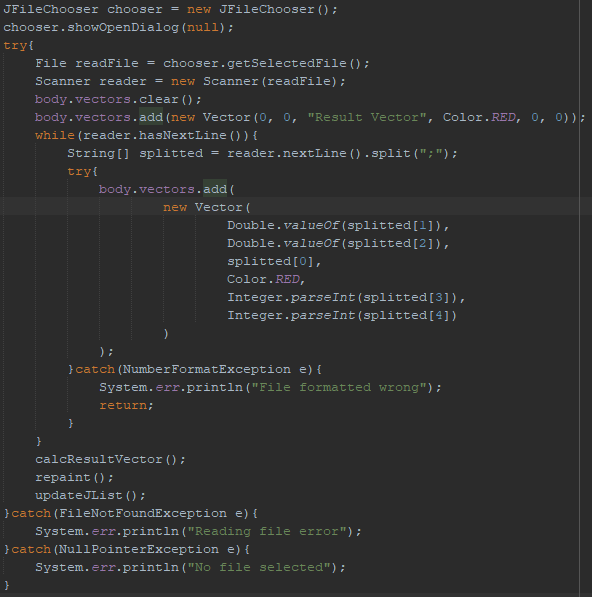
Una volta che l’utente avrà cliccato ok verrà creato un nuovo file, su cui salverò tutti i dati di ogni vettore.

Il modo di rappresentare i vettori all’interno del file è palesemente ispirato al csv.

Nel caso l’utente non scegliesse nessun file verrà stampato un messaggio di errore e non verrà caricato nulla, stessa cosa se ci fosse una IOException

**jUploadActionPerformed**

Questo metodo invece carica un file salvato in precedenza.



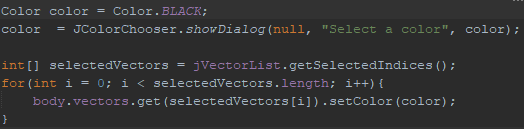
Come nel metodo di save, si apre un popup di JFileChooser.

Quando l’utente clicca ok, si resetta la lista di vettori e si inizia a leggere il file riga per riga, ogni riga viene aggiunto un vettore controllando che i valori siano giustie nel caso non lo fossero stampa un messaggio di errore e ferma il processo.

Infine calcola il vettore risultante, aggiorna la lista di vettori e ridisegna.

**jColorPickerActionPerformed**

In questo metodo l’utente sceglie il colore del vettore selezionato all’interno della jList.



Creo un’istanza di JColorChooser che fa apparire un popup, quando l’utente sceglie il colore percorro la lista di vettori selezionati e setto il colore scelto.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-01 | **Nome:** | Test disegno vettori |
| **Descrizione:** | verificare che la parte sinistra accetti input e crei il vettore tramite mouse | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Cliccare sulla parte vuota della gui a sinistra | | |
| **Risultati attesi:** | Nella GUI viene tracciata una linea dal punto rosso fino a dove si è cliccato | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-01 | **Nome:** | Test GUI responsive |
| **Descrizione:** | Verificare che la GUI sia responsive | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Ridimensionare la finestra | | |
| **Risultati attesi:** | In orizzontale aumenta di dimensione il canvas dove vengono disegnati i vettori, in verticale invece la lista di vettori | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-02 | **Nome:** | Test creazione vettori |
| **Descrizione:** | Verificare che il vettore venga creato, che venga rappresentato nella lista, nel canvas e che vengano rivelati i vettori quando si seleziona il vettore dalla lista | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Creare un vettore (tramite mouse o tramite textbox) 2. Verificare che si sia aggiunto un vettore nella lista di vettori e cliccarci sopra | | |
| **Risultati attesi:** | Nella sezione degli input verranno mostrati i valori del vettore selezionato nella lista, e verrà disegnato sul canvas. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-02 | **Nome:** | Test nascondere vettori |
| **Descrizione:** | Verificare che i vettori vengano nascosti cliccando il tasto hide | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Creare qualche vettore 2. Cliccare il tasto Hidden in alto a destra (di fianco a delete vector) | | |
| **Risultati attesi:** | Nel canvas i vettori spariranno, e se si riclicca lo stesso bottone riappariranno | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-02 | **Nome:** | Test colorare vettori |
| **Descrizione:** | Verificare che i vettori possano cambiare colore in base alla scelta dell’utente | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Creare un vettore 2. Selezionare il vettore appena creato 3. Cliccare il tasto “Color” 4. Quando si apre il popup, selezionare un colore e cliccare “ok” | | |
| **Risultati attesi:** | Il popup si chiuderà e il vettore selezionato sarà del colore scelto | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-02 | **Nome:** | Test modificare vettori |
| **Descrizione:** | Verificare che i vettori possano cambiare valori e che essi vengano modificati nella rappresentazione | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Creare un vettore 2. Selezionare il vettore appena creato 3. Modificare un valore negli input 4. Cliccare “New Vector” | | |
| **Risultati attesi:** | Il vettore selezionato, cambiandogli valori, cambierà anche la sua rappresentazione nel canvas in base ad essi. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-03 | **Nome:** | Test animazione |
| **Descrizione:** | Verificare che l’animazione funzioni, che segua il vettore risultante e che lasci la scia | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Creare uno o più vettori 2. Cliccare il tasto start sotto al canvas | | |
| **Risultati attesi:** | Il punto rosso comincia a muoversi seguendo il vettore risultante, lasciando dietro di sé una riga viola | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-02  REQ-03 | **Nome:** | Test cambiamento durante l’animazione |
| **Descrizione:** | Verificare che durante l’animazione uno o più vettori possano cambiare i loro valori, il valore di questo cambiamento si setta quando si crea il vettore | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Creare un vettore con la x variabile diversa da 0 2. Far partire l’animazione | | |
| **Risultati attesi:** | il vettore risultante cambia nel tempo e il corpo segue il vettore | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-05 | **Nome:** | Test scarica dati |
| **Descrizione:** | Verificare che scaricando i dati si possa accedere al file | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Creare uno o più vettori 2. Cliccare il tasto “save” 3. Una volta aperto il popup, scegliere la destinazione e il nome del file 4. Dopo aver cliccato ok, andare a controllare se il file è stato creato | | |
| **Risultati attesi:** | Il file è stato creato con all’interno le informazioni dei vettori separate da ‘;’ | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-05 | **Nome:** | Test Upload dati |
| **Descrizione:** | Verificare che scaricando i dati si possa accedere al file | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Cliccare il tasto “upload” 2. Una volta aperto il popup, scegliere il file contenente le informazioni dei vettori 3. Cliccare ok | | |
| **Risultati attesi:** | I vettori importati sono quelli che ci si aspettava che fossero e appaiono nel canvas e nella lista | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-04 | **Nome:** | Test resizing |
| **Descrizione:** | Verificare che l’utente possa allontanare la visuale | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Girare la rotellina del mouse | | |
| **Risultati attesi:** | la schermata si rimpicciolisce e si vede tutto zoommato all’indietro | | |

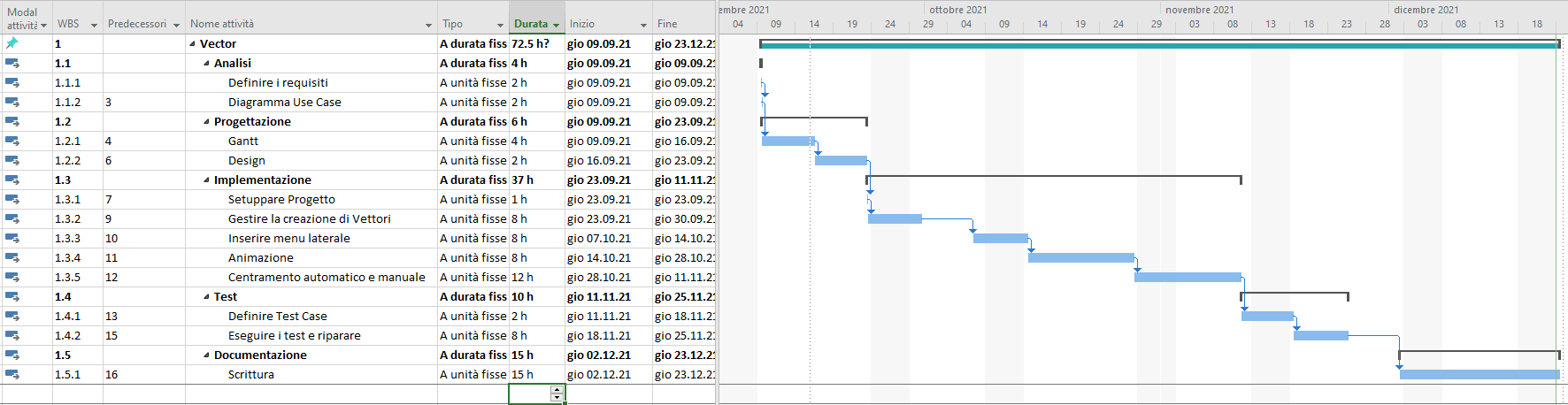
## Risultati test

Tutti i test sono riusciti, tranne il TC-011, c’è qualche problema con il resizing che non sono riuscito a risolvere.

## Mancanze/limitazioni conosciute

Il resizing non funziona per via delle conoscenze che non sono riuscito ad acquisire sui JFrame durante il corso.

# Consuntivo

Rispetto alla pianificazione iniziale ci ho messo relativamente poco ad implementare il codice, ma molto di più a fare la documentazione e a riparare errori.

Questo è dato dal fatto che non ho saputo gestire bene il tempo e sono arrivato tirato alla fine senza aver fatto molti sforzi una volta finito di programmare il simulatore.

# Conclusioni

Non credo che questo progetto sia stato una grande perdita di tempo, mi sono divertito a programmare e mi ha interessato il tema e l’implementazione; tuttavia, devo dire che questo applicativo non ha un impatto rilevante e non è effettivamente utile.

## Sviluppi futuri

Probabilmente questo progetto non avrà nessuno sviluppo futuro e non verrà mai aggiornato, ma logicamente si potrebbe aggiungere una dimensione, migliorando il sistema di movimento della visuale sugli assi.

Non credo sia necessario aggiungere altro o pensare di aggiungere altro.

## Considerazioni personali

In questo progetto posso dire che ho imparato che il tempo non è mai abbastanza, e quando pensi di averne abbastanza non ti rendi conto del fatto che non è sufficiente.

Effettivamente ho procrastinato fino a quando non mi sono reso conto che non avevo più tempo, e lì ho dovuto fare molte cose di fretta maledicendomi per il mio mancato impegno.

Tralasciando la mia scarsa capacità di gestione del tempo, posso dire che il progetto è stato arricchente e che sono felice del lavoro che ho fatto, nonostante non sia il massimo che potevo fare, anzi se vogliamo essere sinceri per quanto riguarda la documentazione e i diari c’è molto da ridire.

Il fatto che questo applicativo non abbia nessuno scopo e nessun’utilizzo effettivo mi rattrista, non è molto bello quando ti impegni per qualcosa e questa cosa è inutile.

Non dico che questo applicativo sia totalmente inutile, effettivamente potrebbe velocizzare il calcolo, ma resta comunque un calcolo semplice e l’animazione non mi pare di grande utilizzo.

Credo comunque che come apprendimento didattico mi abbia dato molto.

# Allegati

Elenco degli allegati:

* Diari di lavoro
* Codici sorgente
* Mandato e/o QdC
* Prodotto