Documentazione del progetto di simulazione di forze vettoriali

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Use case 5

1.7 Pianificazione 6

1.8 Analisi dei mezzi 6

1.8.1 Software 6

1.8.2 Hardware 6

2 Progettazione 7

2.1 Design delle interfacce 7

2.2 Design dei dati e database 7

3 Implementazione 8

3.1 interfaccia 8

3.2 Classe Vector 8

3.3 Creazione vettori 9

3.3.1 graficamente: 9

3.3.2 inserendo i valori: 9

3.4 Metodi per aggiornare l’interfaccia 9

3.5 Modo per vedere i dati dei vettori 10

3.6 Altri metodi utili 10

3.7 grandi problemi e soluzione 10

3.8 Colore dei vettori 10

3.9 Rimozione vettori 10

3.10 Simulazione forza 10

3.11 Save e import dei dati 11

3.12 Possibilità di modificare i vettori 11

4 Test 11

4.1 Protocollo di test 11

4.2 Risultati test 14

4.3 Mancanze/limitazioni conosciute 14

5 Consuntivo 14

6 Conclusioni 14

6.1 Sviluppi futuri 14

6.2 Considerazioni personali 15

6.3 Sitografia 16

7 Allegati 16

# Introduzione

## Informazioni sul progetto

* Allievo: Andrea Frati con ruolo di produttore
* Supervisore: Geo Petrini con ruolo di cliente
* SAM Trevano, sezione informatica, terzo anno, modulo 306
* data di inizio progetto: 09.09.2021
* data di consegna: 23.12.2021

## Abstract

Per calcolare la forza risultante avendo dei vettori e per passare da coordinate polari a coordinate cartesiane bisogna conoscere le formule necessarie e saperle utilizzare, e probabilmente serve anche disegnare i vettori per poter visualizzare la situazione. Questo programma serve ad avere un modo semplice e con un’interfaccia grafica per calcolare la forza risultante, creare i vettori e potere vedere la situazione graficamente con anche la possibilità di vedere una simulazione della forza in azione. È quindi molto utile per guadagnare tempo durante un lavoro con i vettori ed è comodo per vedere bene le forze e che effetto hanno se messe in azione, con il vantaggio di poter salvare una simulazione per poi continuarla in un secondo momento.

## Scopo

Lo scopo del progetto è quello di creare un’interfaccia grafica utile a calcolare la forza risultante dei vettori (creati tramite inserimento dati o graficamente) e a convertire coordinate polari in coordinate cartesiane e viceversa. L’obiettivo è anche quello di poter mostrare graficamente la situazione che si vuole creare con anche una simulazione dell’azione della forza risultante e la possibilità di salvare e caricare delle simulazioni.

## Analisi

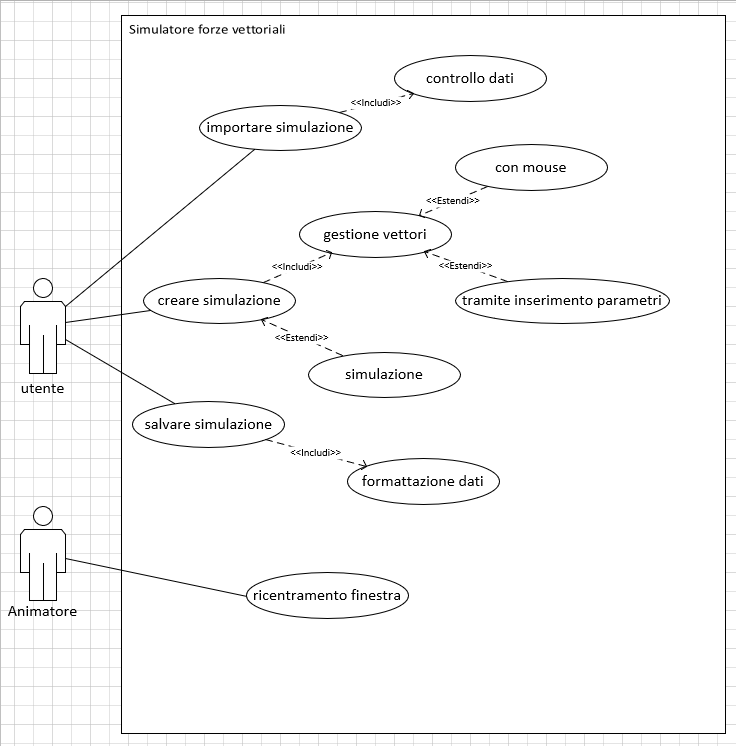
## Analisi del dominio

Questo prodotto potrà essere utilizzato su qualsiasi sistema operativo e su qualsiasi pc, l’unico requisito della macchina utilizzata è quello di dover avere java installato. Per poter lanciare il programma serve sapere dei comandi minimi, basta entrare con il cmd nella cartella dove è situato il file SimulatoreDiForzeVettoriali.jar e scrivere il comando java -jar SimulatoreDiForzeVettoriali.jar e così facendo apparirà l’interfaccia grafica del progetto.

## Analisi e specifica dei requisiti

|  |  |  |
| --- | --- | --- |
| id | descrizione | priorità |
| 1 | interfaccia GUI ridimensionabile | 1 |
| 2 | oggetto che si muove in base alle forze dei vettori | 1 |
| 3 | Definizione del vettore tramite mouse | 2 circa |
| 4 | Disegnare i vettori in base a dei punti dati dall’utente | 1 |
| 5 | Scelta colore vettore da parte dell’utente | 3 |
| 6 | Vedere e poter modificare i parametri dei vettori | 2 |
| 7 | Visualizzare la risultante in tempo reale | 1 |
| 8 | Possibilità di avviare stoppare e azzerare la simulazione in qualsiasi momento | 2 |
| 9 | disegnare la scia lasciata dall’oggetto in movimento | 3 |
| 10 | Gestire se l’oggetto esce dai bordi | 1 |
| 11 | Possibilità di salvare e caricare i parametri della simulazione | 2 |

## Use case



## Pianificazione

Modello di pianificazione: waterfall

|  |
| --- |
| Figura 2: Esempio di diagramma di Gantt. |

## Analisi dei mezzi

Per la realizzazione del progetto è stato utilizzato il computer messo a disposizione dalla scuola con installati netbeans e java.

### Software

netbeans: versione 12.4

Sistema operativo: Windows 10 Enterprise versione 20H2 (build So 19042.870)

Java: versione 16.0.1

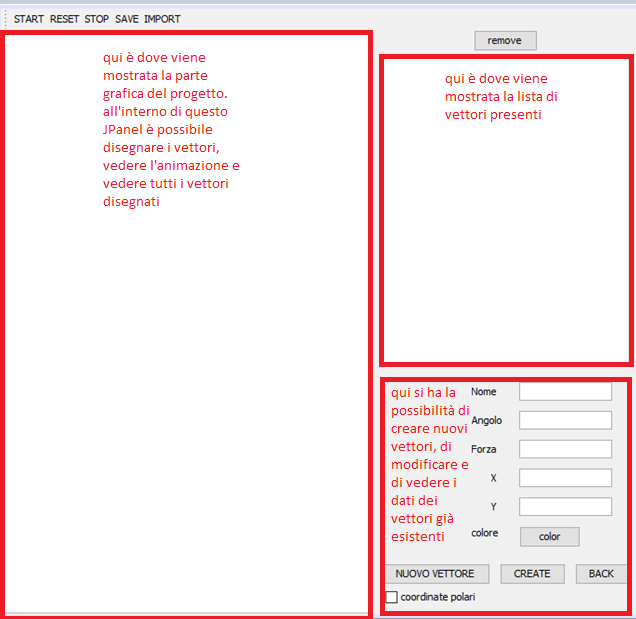
### Hardware

Processore: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz

RAM: 16 GB

# Progettazione

## Design delle interfacce



## Design dei dati e database

Non c’è un database ma c’è la possibilità di salvare dati in file di testo.

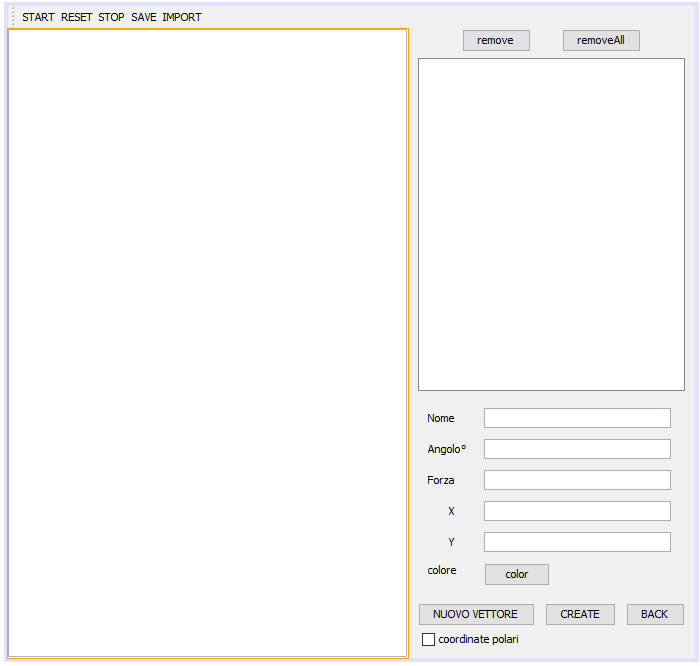
Quando si schiaccia sul pulsante salva, i dati vengono salvati in un file di testo con la data attuale come nome, ogni riga descrive i dati di un vettore separati da un punto e virgola.

Il file viene salvato nella stessa cartella di dove si trova il file SimulatoreDiForzeVettoriali.jar

# Implementazione

## interfaccia

La prima cosa che ho fatto è stata quella di creare un’interfaccia grafica usando netbeans già provando a prevedere tutti i bottoni e textbox che avrei avuto bisogno per tutto il progetto anche se ho dovuto fare alcune piccole aggiunte durante il progetto.



## Classe Vector

Un vettore è definito dalla classe Vector. Un vettore ha una x, una y, un angolo, una forza, un nome, e un colore. La classe possiede un costruttore che riceve il nome del vettore, due parametri double e un parametro boolean che definisce se i due double sono x e y oppure forza e angolo.

Il costruttore a dipendenza dei dati che riceve calcola i dati che gli mancano usando i metodi di calcolo presenti nella classe ovvero:

calcolaForza(double x, double y)

calcolaX(double x, double y)

calcolaY(double angolo, double forza)

calcolaAngolo(double angolo, double forza)

ci sono anche dei metodi set che servono a ricalcolare i valori che devono essere ricalcolati quando viene cambiato un valore del vettore.

## Creazione vettori

I vettori possono essere creati in due modi:

### graficamente:

quando premo un tasto del mouse e sono all’interno del JPanel (dove vengono disegnati i vettori) e trascino sempre con il mouse premuto, vedrò una riga che si disegna dal punto dove ho premuto al punto dove è il mouse. Quando rilascio il tasto del mouse il vettore viene creato usando il costruttore che riceve come parametri x y e nome e calcola forza e angolo e questo vettore viene aggiunto alla lista, il nome è drawVector + un numero che cresce per ogni nuovo vettore creato graficamente es: drawVector1. Il vettore però non viene disegnato dove ho tracciato la linea con il mouse, ma viene collegato a un pallino che poi sarà l’oggetto che si muoverà in base alla forza risultante.

Per poter implementare questa funzionalità ho usato gli eventi mousePressed, mouseDragged e mouseReleased.

mousePressed: prendo la coordinata del punto in cui premo il tasto sinistro del mouse.

mouseDragged: per ogni movimento del mouse disegno una riga da dove ho premuto il mouse a dove è il cursore

mouseReleased: prendo le coordinate del punto in cui rilascio, calcolo la x e la y facendo la differenza tra le coordinate di pressed e released, creo il vettore, aggiungo il vettore alla lista, aggiorno la lista graficamente, ricalcolo la forza risultante e aggiorno il disegno.

### inserendo i valori:

c’è un checkBox che permette di decidere se creare un vettore dando x e y oppure angolo e forza.

Quando premo il bottone nuovo vettore appaiono solo i textbox nome e o x e y oppure angolo e forza a dipendenza se il checkBox è selezionato o no. Appena inseriti i dati si può premere il tasto create per creare il vettore, il tasto back serve per tornare indietro se ho sbagliato a selezionare il checkBox.

Il tasto che ha la funzione principale è il tasto create che fa più o meno lo stesso lavoro del mouseReleased quindi crea il vettore prendendo i dati nei textboox e lo aggiunge alla lista aggiornando tutto graficamente ma in più fa anche i controlli sui parametri inseriti e decide in base ai dati che riceve se usare il costruttore usando x e y oppure angolo e forza.

## Metodi per aggiornare l’interfaccia

Ho citato più volte il fatto di aggiornare la lista graficamente, la forza risultante e i disegni dei vettori e per fare questo ci sono tre principali metodi.

Il metodo paint si occupa di disegnare sul JPanel, aiutandosi con dei metodi di supporto disegna i vettori presenti nella lista di vettori, il pallino e la linea che segue il pallino, la griglia sullo sfondo e si occupa di gestire il disegno del mouseDragged. Poi c’è il metodo updateList che aiutandosi con il metodo addVectorNames prende tutti i nome dei vettori e li aggiunge alla JList , e infine c’è il metodo updateRisOrigin che fa passare tutti i vettori della lista e calcola la forza risultante.

## Modo per vedere i dati dei vettori

Per poter vedere i dati di un vettore basta selezionarlo graficamente nella JList e i textbox verranno riempiti con i dati di quel vettore. Questo si può fare semplicemente prendendo l’indice del vettore selezionato graficamente e prendendo i dati del vettore che si trova nella lista a quella posizione. Infine si riempiono i textbox con i valori di quel vettore. Per fare questo c’è un metodo writeVectorInformation che riceve come parametro un vettore e si occupa di scrivere i suoi dati nei textbox.

## Altri metodi utili

Ci sono anche degli altri metodi di supporto per la grafica del progetto:

allVisible: che rende visibili i textbox

allInvisible: che rende invisibili i textbox

setEmptyText: che svuota il contenuto di tutti i textbox

e c’è un metodo che serve per i controlli dei valori per la creazione dei vettori

isDouble: ritorna true se il valore passato è double

## grandi problemi e soluzione

un grande problema riscontrato è il fatto che i vettori devono lavorare su un piano cartesiano e quindi aspettandosi che inizialmento il pallino è alle coordinate 0,0. Invece le coordinate 0,0 sono in alto a sinistra e per mettere a posto questa cosa bisogna cambiare il segno della y quando si muove il pallino e il normale calcolo dell’angolo non funziona nel modo corretto e quindi bisogna aggiustare i risultati usando un metodo fixAngle che ricalcola gli angoli nel modo giusto.

## Colore dei vettori

Vicino ai textbox con i valori dei vettori, c’è un bottone chiamato color che serve per cambiare il colore ai vettori, questo non è possibile farlo durante la creazione del vettore ma solo in un secondo momento selezionandolo sulla lista. Questo bottone fa apparire una finestra con un JColorChooser in cui è possibile scegliere un colore, questo colore verrà poi assegnato all’attributo color presente nella Classe Vector.

Poi viene chiamato il repaint che quando disegna i vettori setta il colore in base all’attributo del vector, il bottone color viene colorato con il colore del vettore selezionato così c’è anche la possibilità di poter distinguere di quale vettore si stanno guardando le informazioni.

## Rimozione vettori

C’è anche la possibilità di rimuovere dei vettori dalla lista con il bottone remove che rimuove il vettore selezionato dalla lista e che usa i metodi updateList e repaint per aggiornare tutto in base alla nuova lista senza il vettore rimosso. Questo bottone non funziona se provo a rimuovere la forza risultante e se provo a rimuovere l’ultimo vettore rimasto. Per farlo guardo gli indici selezionati e se c’è solo un indice selezionato, se l’indice selezionato non è la forza risultante e se la grandezza della lista è maggiore di due allora rimuovo il vettore dalla lista e richiamo repaint e updateList.

C’è inoltre il bottone removeAll che si occupa di riportare tutto alla situazione iniziale svuotando la lista dei vettori, mettendo il pallino al centro e svuotando l’array che permette di disegnare la linea che segue il pallino e poi richiamando tutti i metodi per ridisegnare sulla finestra.

## Simulazione forza

Per la parte di simulazione della forza invece ci sono 3 bottoni principali che fanno funzionare il tutto.

Inizialmente c’è un pallino al centro dello schermo con collegati tutti i vettori. Quando avvio la simulazione premendo il tasto start il pallino inizia a muoversi nella direzione della forza risultante con una velocità proporzionata in base ai dati della forza risultante. Per farlo avvio un timer che ogni millisecondo cambia le variabili che contengono la posizione del pallino aumentandole di xForzaRis / 150 e yForzaRis / 150 e chiamo il repaint per aggiornare il disegno.

Poi quando il pallino è in movimento posso schiacciare il tasto stop che ferma la simulazione lasciando il pallino li dov’è. Per farlo ho semplicemente fermato il timer e chiamato il repaint. Poi c’è la possibilità di resettare la simulazione fermando il timer, assegnando la posizione del centro del JPanel alle variabili che indicano la posizione dell’oggetto e chiamando il repaint.

## Save e import dei dati

Un’altra funzione del programma è quella di poter salvare e importare simulazioni per questo esistono i bottoni save e remove. Il bottone save crea un file con la data del salvataggio all’interno del nome e fa passare tutta la lista di vettori tranne la forza risultante. Per ogni vettore crea una stringa con i suoi dati (x, y, nome) + la posizione x e y del pallino e i punti dell’array di linee che disegna la linea che segue il pallino, e scrive tutto questo dentro al file creato. Poi per importare una situazione creata in precedenza c’è il tasto import che fa apparire un JFileChooser dove si può selezionare il file da importare e quando viene selezionato, viene eliminata tutta la lista della situazione attuale e viene ricreata la nuova lista leggendo i valori dal file e ricreando tutti i vettori, ricalcolando la forza risultante e ridisegnando tutto nel posto di prima. Il pallino viene ridisegnato nella posizione di quando è stata salvata la simulazione leggendo x e y salvati sul file e viene riempito di nuovo l’array di punti per disegnare la linea.

## Possibilità di modificare i vettori

C’è anche la possibilità di cambiare i valori dei vettori in tempo reale selezionando il vettore che si desidera cambiare e modificando i valori dei textbox. Per far si che funzioni in tempo reale ascolto ogni cambiamento di ogni textbox usando l’evento careUpdate e quando c’è un cambiamento ricalcolo tutto e ridisegna tutto in base al dato cambiato, ovviamente bisogna stare attenti a far si che ricalcoli tutto solo quando cambio i valori di un textbox usando la tastiera e non quando per esempio seleziono un altro vettore e quindi il programma cambia i dati scritti nel textbox, per fare questo ho usato una variabile booleana che mi dice quando sto modificando i dati a mano e quindi quando usare il careUpdate.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-01 | **Nome:** | GUI |
| **Descrizione:** | Creazione interfaccia GUI | | |
| **Procedura:** | Far partire il programma e controllare che si veda la GUI e che si possa muovere | | |
| **Risultati attesi:** | Apertura di una finestra con la GUI | | |
| **Test Case:**  **Riferimento**: | TC-002  REQ-03  REQ-04 | **Nome:** | Disegno vettori |
| **Descrizione:** | Testare I due modi per disegnare i vettori | | |
| **Procedura:** | 1. Provare a disegnare con il mouse sul canvas e controllare che venga disegnato il vettore esattamente dove sono passato con il mouse. 2. Provare a inserire con la tastiera dei valori di vettori e controllare che vengano disegnati nel posto giusto | | |
| **Risultati attesi:** | Due metodi funzionanti per inserire vettori che si vedano graficamente | | |
| **Test Case:**  **Riferimento**: | TC-003  REQ-02  REQ-08 | **Nome:** | Muovere oggetto |
| **Descrizione:** | Testare che l’oggetto si muova nella direzione e con la forza desiderata | | |
| **Procedura:** | 1. Mettere un bottone da schiacciare per far partire la simulazione 2. Schiacciarlo e controllare che l’oggetto si muova nella direzione giusta | | |
| **Risultati attesi:** | Quando il bottone viene cliccato l’oggetto fermo al centro dello schermo inizoia a muoversi nella direzione della forza risultante | | |
| **Test Case:**  **Riferimento**: | TC-004  REQ-10 | **Nome:** | ridimensionamento |
| **Descrizione:** | Quando si esce dal bordo del canvas l’interfaccia si ridimensiona | | |
| **Procedura:** | 1. Far partire la simulazione e aspettare che l’oggetto si muova fino ad uscire dal bordo del canvas 2. Appena l’oggetto esce dai bordi il canvas dovrebbe ridimensionarsi | | |
| **Risultati attesi:** | Un canvas che si ridimensiona per far visualizzare l’oggetto che si allontana dal centro | | |
| **Test Case:**  **Riferimento**: | TC-005  REQ-11 | **Nome:** | Salvataggio e caricamento parametri |
| **Descrizione:** | Implementare un modo per poter caricare dei dati presi in precedenza o di salvare quelli appena inseriti | | |
| **Procedura:** | 1. Creare una situazione con un paio di vettori 2. Usare l’opzione di salvare I dati 3. Chiudere l’applicazione oppure eliminare tutti I dati inseriti 4. Provare a caricare I dati appena salvati 5. Se c’è un errore creare un file con dei dati salvati che sono sicuramente giusti e provare a caricarlo per vedere se è l’upload o il download che non funziona | | |
| **Risultati attesi:** | Dati salvati in modo corretto e dati caricati in modo corretto | | |
| **Test Case:**  **Riferimento**: | TC-006  REQ-08 | **Nome:** | Avvio pausa e azzeramento simulazione |
| **Descrizione:** | Possibilità di mettere in pausa e azzerare la simulazione dopo che è stata avviata | | |
| **Procedura:** | 1. Avviare la simulazione con il tasto già presente da un requisito precedente e creare un tasto stop e uno reset 2. Provare a usare il tasto stop mentre l’oggetto si muove e controllare se si ferma 3. Provare a usare il tasto reset e controllare se effettivamente l’oggetto torna al punto di partenza | | |
| **Risultati attesi:** | Tasti di stop e di reset funzionanti e cliccabili in qualsiasi momento | | |
| **Test Case:**  **Riferimento**: | TC-007  REQ-05 | **Nome:** | Cambiare colore dei vettori |
| **Descrizione:** | Possibilità di decidere di che colore è ogni vettore | | |
| **Procedura:** | 1. aprire la schermata di visualizzazione dei parametri e provare a cambiare il colore dell’input color 2. provare a mettere un colore diverso a ogni vettore | | |
| **Risultati attesi:** | Il colore dei vettori cambia appena viene definite nell’input color | | |
| **Test Case:**  **Riferimento**: | TC-008  REQ-09 | **Nome:** | Disegnare la scia di spostamento dell’oggetto |
| **Descrizione:** | L’oggetto quando si muove deve lasciare dietro di se una scia per poter capire dove è passato | | |
| **Procedura:** | 1. avviare la simulazione e controllare se dove passa l’oggetto resta una scia 2. provare a resettare la simulazione e controllare che la scia scompaia | | |
| **Risultati attesi:** | L’oggetto quando si muove lascia una scia dietro di se | | |

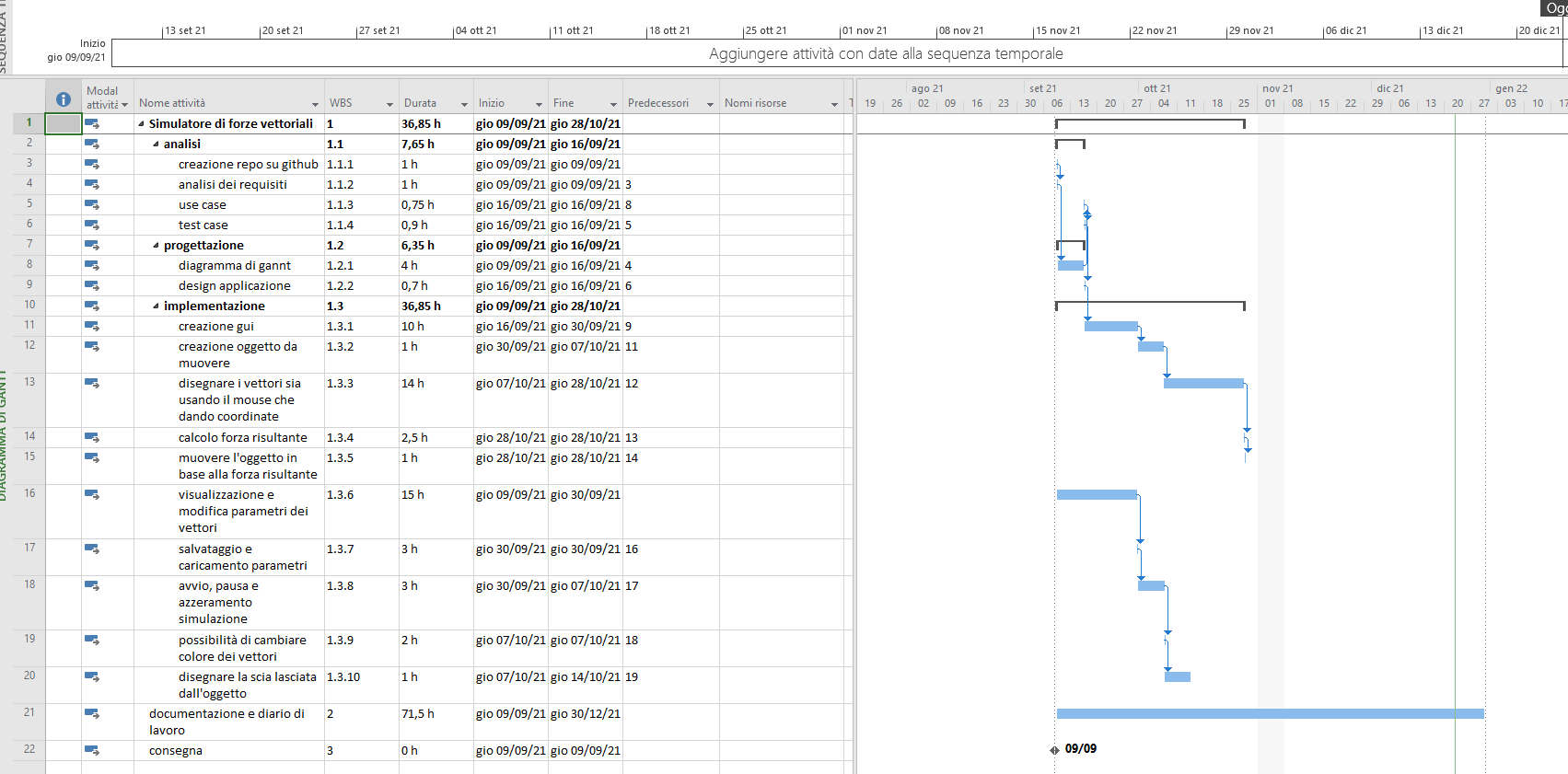
## Risultati test

|  |  |
| --- | --- |
| TC-001 | All’avvio del progetto si apre una finestra con l’interfaccia grafica del progetto |
| TC-002 | I vettori si possono disegnare in tutti e due i modi, i dati vengono riportati in modo corretto nella lista di vettori e vengono disegnati rispettando i dati. |
| TC-003 | Quando viene premuto start, l’oggetto si muove nella direzione della forza risultante |
| TC-004 | Non funziona |
| TC-005 | Quando salvo una situazione mi si crea il file con i dati scritti all’interno, e quando impporto la simulazione, tutto ritorna come quando ho salvato. |
| TC-006 | Si può avviare la simulazione, posso stopparla e posso azzerarla |
| TC-007 | Si può cambiare il colore ai vettori |
| TC-008 | La scia viene creata quando la simulazione è attiva e segue esattamente il movimento dell’oggetto |

## Mancanze/limitazioni conosciute

Manca il fatto di poter ridimensionare la finestra.

# Consuntivo



# Conclusioni

Questo progetto è ancora non perfettamente funzionante come vorrei quindi al momento mi sento di dire che è abbastanza inutile, per esempio anche provandoci non sono riuscito a risolvere lo sfarfallio che c’è quando avvio la simulazione e questo fa sì che il programma sia un po’ meno piacevole da utilizzare.

## Sviluppi futuri

Uno sviluppo futuro potrebbe essere quello di gestire oltre a x e y anche z, rendendo il progetto tridimensionale.

## Considerazioni personali

Da questo progetto ho imparato che è molto difficile progettare bene un lavoro dato che è difficile prevedere esattamente tutto quello che c’è da fare. Quindi è anche molto difficile seguire la progettazione dato che magari mentre si sta programmando si preferisce fare prima una cosa a cui non si aveva dato priorità oppure si vuole aggiungere qualcosa al progetto perché ci si accorge che è necessario, e molto altro. Ho imparato anche a usare le interfacce grafiche di java con netbeans e usando bottoni textbox ecc. dato che le mie competenze sulle GUI di java si fermavano a un paio di eventi e il metodo paint. Quindi mi è stato molto utile dato che le interfacce grafiche sono importantissime in quasi qualunque tipo di programma.

## Sitografia

Questi sono i siti usati per cercare di capire il funzionamento dei vettori

* <https://library.weschool.com/lezione/conversione-tra-funzioni-cartesiane-e-coordinate-polari-2196.html>

Questi sono i siti usati per cercare di capire il funzionamento dei componenti grafici di java

* <https://docs.oracle.com/javase/7/docs/api/javax/swing/JList.html>
* <https://docs.oracle.com/javase/tutorial/uiswing/components/colorchooser.html>
* <https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html>
* <https://coderedirect.com/questions/236781/image-flickers-on-repaint>
* https://www.tabnine.com/code/java/classes/java.awt.Graphics2D

Questi sono i siti usati per cercare di capire il funzionamento di altre funzioni utili di java

* https://docs.oracle.com/javase/7/docs/api/java/util/Timer.html

# Allegati

* Diari di lavoro
* Codici sorgente
* documentazione
* QdC