Pika

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 7

1.7 Pianificazione 8

1.8 Analisi dei mezzi 9

1.8.1 Software 9

1.8.2 Hardware 9

2 Progettazione 9

2.1 Struttura file JS 9

3 Implementazione 10

3.1.1 Index 10

3.2 Classi JS 13

3.2.1 Point 13

3.2.2 Rectangle 14

3.2.3 Circle 16

3.2.4 Pencil 17

3.3 Action 19

3.3.1 ActionPoints 19

3.3.2 ActionRects 22

3.3.3 actionCircle 24

3.3.4 actionLines 25

3.4 drawHelp.js 26

3.5 main 27

3.6 imageImport 28

3.7 ImageExport 29

4 Test 30

4.1 Protocollo di test 30

4.2 Risultati test 32

4.3 Mancanze/limitazioni conosciute 33

4.3.1 Strumento secchiello 33

5 Consuntivo 33

6 Conclusioni 33

6.1 Sviluppi futuri 33

6.2 Considerazioni personali 33

7 Glossario 34

8 Allegati 34

# Introduzione

## Informazioni sul progetto

Progetto sviluppato da Gioele Chiodoni sotto supervisione del docente Geo Petrini dal 01.09.2023 al 1.12.2023.

Allievo della Scuola Arti e Mestieri di Trevano nella classe I3BB anno 2023/2024.

## Abstract

*We find ourselves in an era where people are increasingly writing on computers. And because of this they are no longer able to write in beautiful handwriting.*

*This application aims to create fun exercises for children to improve handwriting.*

*To do this, the application requires simplicity of use and intuitiveness.*

*The exercise that allows you to create is an image of dots which, if united, form an image.*

## Scopo

Lo scopo didattico del progetto è quello di imparare a gestire nel modo poi autonomo possibile e ottimale un progetto IT in vista dell’esame pratico a fine quarta.

Mentre lo scopo operativo è quello di creare un’applicazione web o non per poter creare immagini a puntini che, se uniti formano un disegno. L’applicativo deve permettere di importare, disegnare ed esportare le immagini. L’applicazione dovrà essere facile e intuitiva in modo da essere accessibile a chiunque.

Pika ha lo scopo di essere un’applicazione semplice e intuitiva per poter creare esercizi con scopo didattico per i bambini per migliorare la calligrafica e la mobilità della mano.

# Analisi

## Analisi del dominio

Questo applicativo dovrà essere semplice e intuitivo da utilizzare per poter essere accessibile a chiunque abbia un computer. Gli utilizzatori principali per qui è pensato il progetto sono i genitori e i docenti per generare esercizi per i bambini. Questi non dovranno avere conoscenze o praticità con il computer per poter usare il programma.

Grazie a quest’applicazione sarà possibile creare degli esercizi pensati per i bambini di modo da migliorare la mobilità della mano.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Manipolazione puntini |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve poter inserire i puntini a piacimento. |
| **Sotto requisiti** | |
| **001** | Deve anche poter cambiarne l’ordine. |
| **002** | Deve anche poter spostarli. |
| **003** | Deve anche poter eliminarli. |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Strumenti di disegno |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve poter disegnare sull’immagine finale per poter aggiungere parti del disegno. |
| **Sotto requisiti** | |
| **001** | Deve poter usare gli strumenti: Penna, cerchio, rettangolo, secchiello. |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Esportare l’immagine |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Una volta finito di mettere i puntini l’utente deve poter esportare l’immagine con i punti in PNG. |
| **Sotto requisiti** | |
| **001** | Se fattibile anche in formato vettoriale. |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Facilità |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’applicativo deve essere facile da usare |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Soluzione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Deve essere possibile vedere la soluzione del disegno |
| **Sotto requisiti** | |
| **001** | Con e senza l’immagine finale. |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Layer |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Deve poter essere possibile scegliere cosa deve essere visto. |
| **Sotto requisiti** | |
| **001** | Si deve poter nascondere l’immagine |
| **002** | Si deve poter nascondere i puntini |
| **003** | Si deve poter nascondere la soluzione |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Formato |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Le immagini importate devono essere di tipo WEBP, JPEG, PNG |

## Use case

L’unico utente è l’admin che userà l’applicazione. Quest’ultimo per prima cosa potrà solo importare un’immagine. Una volta importata potrà fare diverse azioni: disegnare, manipolare i disegni, ecc. Una volta finito il disegno potrà esportare l’immagine nel formato desiderato.

## Pianificazione

Il progetto è partito il 30.08.203 ed è terminato il 1.12.2023 con la consegna del lavoro.

Durante la pianificazione ho tenuto un margine d’errore per aver margine in caso di imprevisti o nel caso di un giorno di malattia.

Le macrocategorie in qui ho incluso le attività sono:

* Analisi: All’interno di questa categoria ho messo tutto ciò che riguarda la pianificazione, la raccolta dei requisiti e le scelte tecniche iniziali. A questa categoria ho assegnato 1.5 giorni.
* Progettazione: All’interno di questa categoria ho inserito le azioni che riguardano la progettazione tecniche come: design delle classi e il design dell’interfaccia grafica. Per questa categoria ho pianificato 1 giorno.
* Implementazione: All’interno dell’implementazione ho inserito tutte le azioni che riguardano lo sviluppo effettivo dell’applicazione. Per lo sviluppo ho pianificato 7.5 giorni.
* Test: Nei test ho messo tutto ciò che riguarda testare il corretto funzionamento dell’applicazione e i suoi ritocchi necessari. Per questa categoria ho pianificato 1.5 giorni

|  |
| --- |
| Figura 1: Esempio di diagramma di Gantt. |

## Analisi dei mezzi

### Software

Visual Studio Code, WebStorm, HTML 5, CSS 3, JavaScript, Bootstrap 5

### Hardware

Computer fornito dalla scuola e un server FTP.

# Progettazione

## Struttura file JS

Il codice JavaScript lo ho strutturato nei seguenti file:

Nella cartella models ho inserito i file contenenti le classi e i file che permettono di fare azioni di esse.

Mentre fuori dalla cartella model ho messo i file imageImport e imageExport che permettono di importare l’immagine ed esportare i disegni fatti sul canvas.

Poi ho messo anche il main.js che dove controllo le interazioni con i checkbox, radio button, il colore, lo spessore, … In questo file non ci sono metodi che disegnano o servono per disegnare.

Nel file drawHelp.js invece ho messo tutte quelle cose che servono di aiuto a più classi per disegnare. In questo file ci sono gli array che salvano i disegni.

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

# Implementazione

### Index

Lo stile della pagina HTML è stato interamente fatto con Bootstrap 5.

La pagina è tutta dentro un fluid container, che è diviso come mostrato nella seguente immagine.

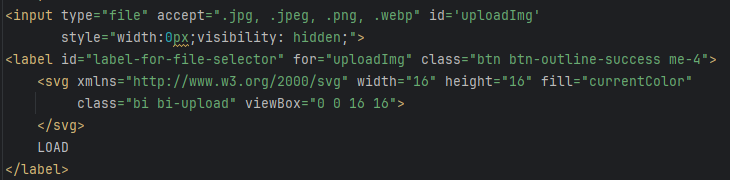
Immagine che contiene schermata, testo, Rettangolo, diagramma

Descrizione generata automaticamente

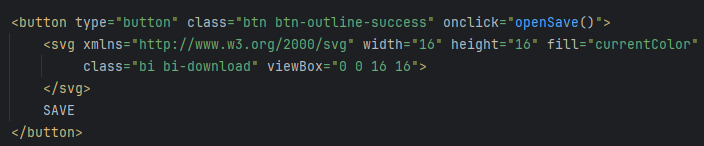
Index.html Container

#### Menu

Il menu è composto dal nome dell’applicazione che è un semplice link a qui ho assegnato la seguente classe di bootstrap: *navbar-band*.

Mentre il bottone LOAD è composto da un input di tipo file a qui ho assegnato un label:

Bottone Load navbar

Il bottone SAVE invece è un semplice bottone:

Nelle seguenti immagini nei tag SVG per motivi pratici è stata rimossa la path.

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamenteQuando si preme il bottone di salvataggio verrà aperto un pannello che contiene le impostazioni di salvataggio. Questo verrà aperto sopra il canvas.

#### Immagine che contiene testo, schermata, Carattere, numero Descrizione generata automaticamente

#### Menu degli strumenti

Il menu che contiene gli strumenti contiene due card, una per gli strumenti e una per i layer.

La scelta degli strumenti avviene tramite degli input di tipo radio.

Mentre la scelta dei layer avviene tramite degli input checkbox.

#### Canvas

Immagine che contiene testo, Carattere, schermata, logo

Descrizione generata automaticamenteAl caricamento della pagina, prima di aver importato un’immagine al posto del canvas verrà mostrato un messaggio d’informazione.

Una volta selezionata l’immagine questo verrà reso invisibile, e verrà reso visibile il canvas con l’immagine di sfondo.

Per far sì che i disegni non escano sgranati o deformati quando vengono disegnati sul canvas ho dovuto assegnare le seguenti proprietà CSS al canvas:

#### Menu di modifica

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Il menu per modificare i disegni è anch’esso come il menu degli strumenti composto da un a card contenente una lista.

Nella lista ho inserito il bottone delete, un input type color, un input type number e un input type range.

## Classi JS

I disegni sono rappresentanti come classi.

Ogni disegno una volta fatto viene salvato nel suo array contenete tutti i disegni fatti dallo stesso strumento. Questi array si trovano nel file drawHelp.js.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

### Point

Immagine che contiene testo, schermata, Carattere, schermo

Descrizione generata automaticamenteLa classe Point rappresenta i puntini numerati.

Per poter istanziare un Point il costruttore richiede un event per poterne ricavare le coordinate, il canvas per poter far il calcolo delle coordinate, il suo numero e il colore.

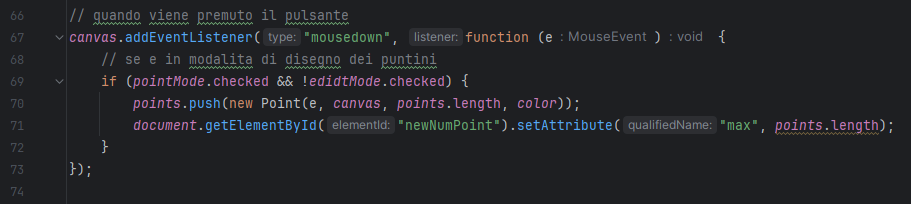
Una volta istanziato il costruttore richiamerà automaticamente la funzione drawPoint che permette di disegnare il punto.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteNel caso in qui il puntino è selezionato verrà disegnato con un’ombreggiatura rossa.

La funzione reDraw si occupa semplicemente di richiamare la funzione drawPoint. Quest’ultima è stata fatta per far si che ogni classe abbia una funzione reDraw (non necessaria).

#### Event

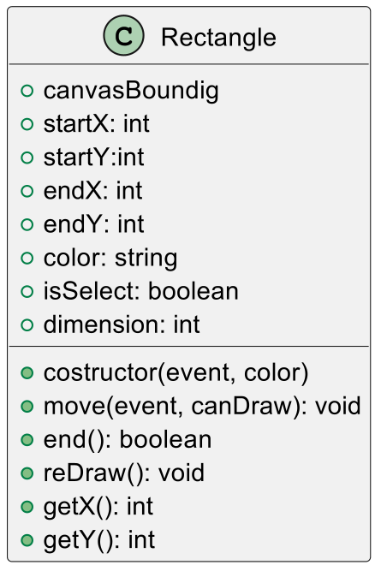
Per catturare l’evento che fa disegnare il puntino ho creato questa funzione anonima

Questa verrà chiamata al click del mouse.

Il Point verrà istanziato sono nel caso in qui l’utente avrà selezionato lo strumento Point e non sarà nella modalità di modifica. Se questa condizione si avvera allora aggiunto un nuovo punto nell’array ’points’.

### Rectangle

La classe Rectangle rappresenta i rettangoli.



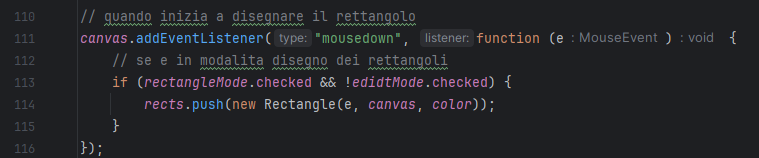
Per poter istanziare un rettangolo è necessario passargli un event e un color nel costruttore.

L’event servirà per poter assegnare le coordinate startX e startY che definiscono il punto in qui parte il rettangolo. Mentre endX e endY servono per definire la larghezza e l’altezza del rettangolo.

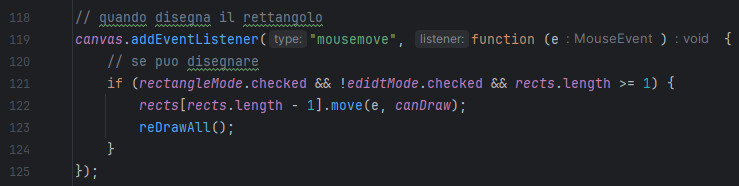
Le funzioni getX e getY servono per poter calcolare le coordinate.

#### Event

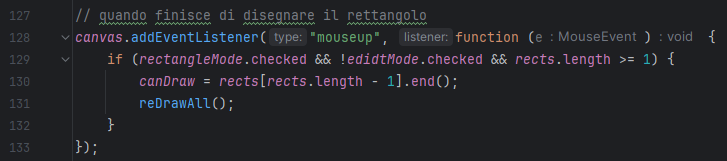
Per poter iniziare a disegnare un rettangolo ho creato questa funzione anonima che se l’utente si trova in modalità rettangolo e con la modalità modifica disattivata aggiunge all’array rects una nuova istanza dei Rectangle.



Poi se l’utente tenendo premuto il mouse lo trascina verrà richiamata la seguente funzione anonima che prenderà l’ultima istanza nell’array e richiamerà la funzione move e continuerà a ridisegnare il rettangolo finche non molla il mouse.

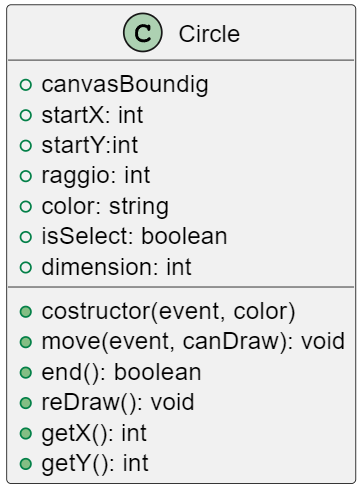


A questo punto una volta aver rilasciato il mouse imposterà canDraw a false di modo che non possa più disegnare.



### Circle

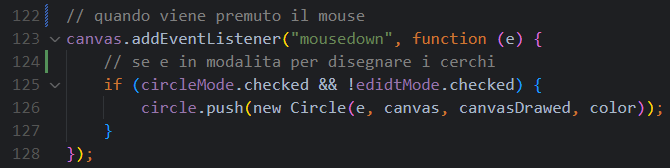
La classe Circle serve per disegnare i cerchi.

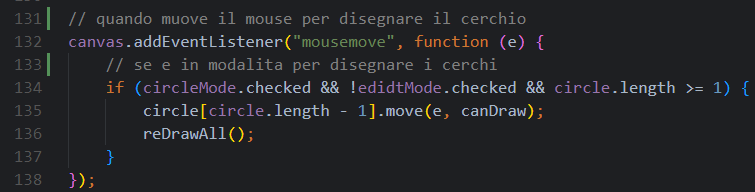
Il funzionamento della classe Circle è uguale alla classe Rectangle.

L’unica differenza è che al posto di avere delle coordinate di inizio e di fine ha solo una coordinata che rappresenta il centro del cerchio e il suo raggio.

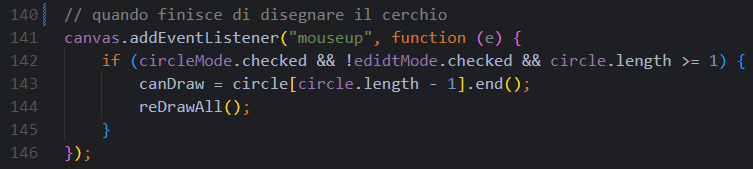
#### Event

Anche qui per disegnare un cerchio ho creato tre funzioni anonime.

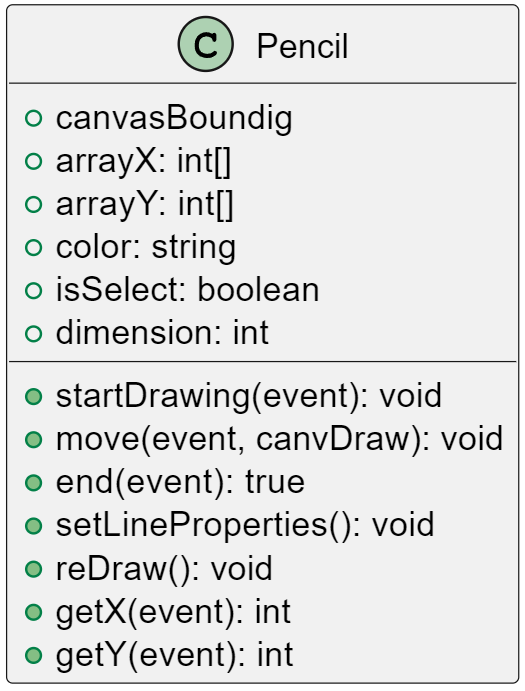
La prima serve per quando inizia a disegnare il cerchio.

La seconda per quando muove il mouse per disegnarlo definendogli la lunghezza del raggio

Mentre la terza per quando finisce di disegnare il cerchio rilasciando il mouse.



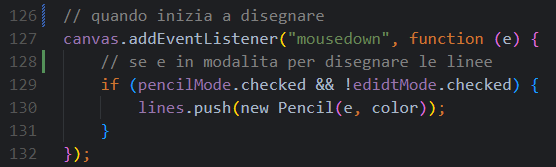
### Pencil

La classe Pencil rappresenta le linee.

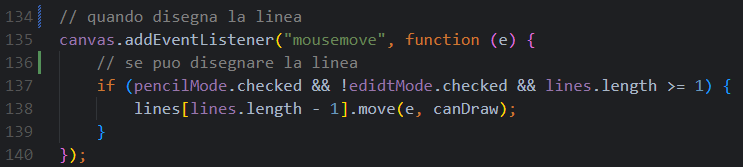
Una linea è composta da tante coordinate che rappresentano lo spostamento del mouse. Queste vengono salvate negli arrayX e arrayY e vengono aggiunte nella funzione move che viene chiamata nella seconda funzione anonima che viene richiamata allo spostamento del mouse. Ad ogni spostamento del mouse quest’ultima richiamerà move passandogli la posizione del mouse che poi verranno aggiunti agli array.

La funzione reDraw per ridisegnare la linea ripercorrerà gli array tracciando le linee da coordinata a coordinata.

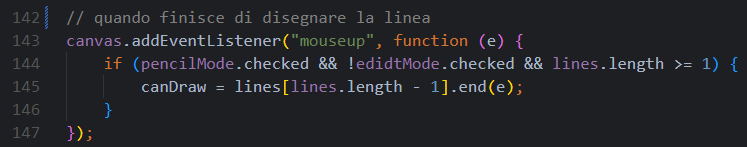
#### Event

Per iniziare a disegnare la linea ho creato questa funzione anonima che verrà richiamata alla pressione del mouse.

Quando inizia a disegnare nell’array lines verrà istanziata una nuova Pencil che rappresenterà la linea.

Poi ho creato una seconda funzione anonima per quando trascina il mouse tracciando la linea.

Questa richiamerà la funzione move.

E per finire di disegnare ho creato un'altra funzione anonima che viene chiamata al rilascio del mouse.

## Action

I file action sono i file che permettono di fare azioni sui disegni (oggetti).

### ActionPoints

Questo file permette di fare azioni sui Point.

#### Selezione

Per Prima cosa per poter fare delle azioni su un puntino bisognerà selezionarne uno. Questo viene fatto tramite il doppio click con questa funzione anonima.

Per prima cosa aggiorno le variabili che permettono il calcolo delle coordinate.

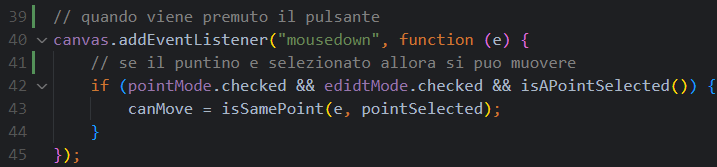
Poi controllo se l’utente ha selezionato lo strumento disegno Point ed è nella modalità edit.

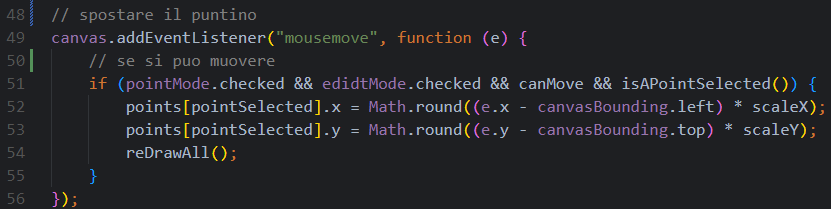
Poi se si trova nella modalità giusta controllo che il click sia avvenuto vicino ad un puntino tramite la funzione getPointClicked(e) che ritorna la posizione del puntino nell’array oppure null se non c’è nessun punto vicino.

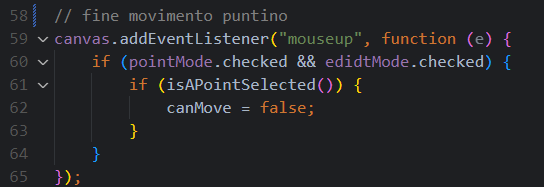
Nel caso in qui il puntino ritornato sia già selezionato allora lo deseleziono, sennò lo seleziono.

#### Spostare

Per spostare un puntino ho dovuto creare queste tre funzioni anonime.

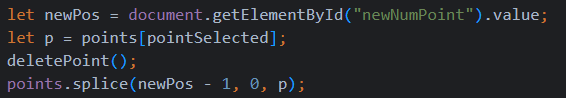
La prima controlla che ci sia un Point selezionato, nel caso in qui esiste allora potrà essere spostato.

Se il puntino può essere spostato allora al movimento del mouse le coordinate del puntino verranno aggiornate con la nuova posizione e poi ridisegno tutto il canvas.

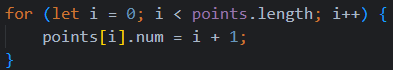
Al rilascio del mouse imposterò la variabile booleana canMove a false di modo da finire lo spostamento.

#### Rinominare

Per rinominare il puntino ho creato questa funzione che viene chiamata al click del bottone da parte all’input type number.

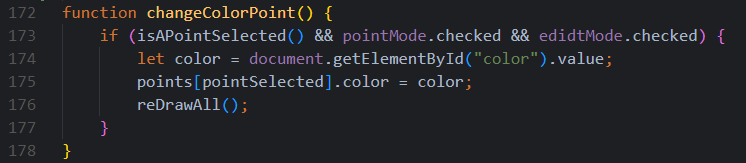
Questa funzione prenderà il nuovo numero del puntino, va nell’array points e prende il puntino selezionato, lo salva e poi lo elimina dopodiché reinserirà il puntino salvato lo reinserisce nella posizione desiderata -1.

In seguito cambierà tutti i numeri dei puntini assegnandogli la propria posizione nell’array points + 1.

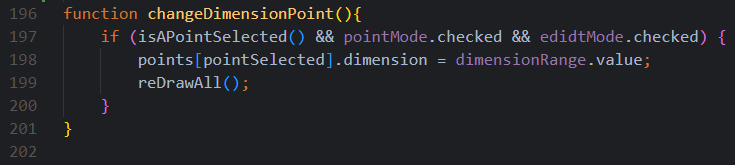


#### Colore

Per cambiare il colore prendo semplicemente il nuovo colore e lo cambio con quello vecchio del puntino selezionato. E poi ridisegno tutto.



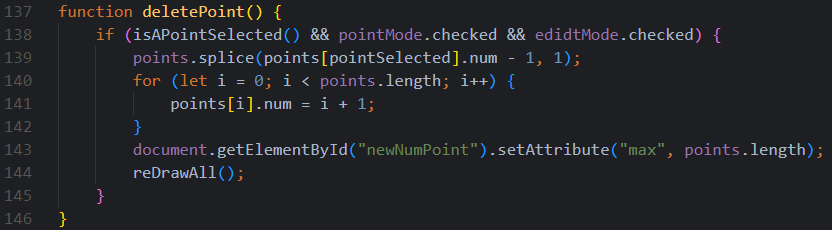
#### Dimensione

Per cambiare la dimensione del puntino semplicemente prendo la nuova dimensione e la imposto al puntino selezionato e poi ridisegno tutto.

#### Delete

Per eliminare un puntino rimuovo il puntino selezionato dall’array points e poi rinomino tutti i puntini con la nuova posizione nell’array + 1.

Per far si che non si possano assegnare numeri più alti ai puntini di quelli esistenti quando si rinomina devo infine cambiare il numero massimo possibile da assegnare.



### ActionRects

Il codice in questo file permette di fare le azioni si rettangoli.

#### Selezionare

Per selezionare i rettangoli ho creato una funzione anonima che viene richiamata con il doppio click.



Prima di poter selezionare un rettangolo controllo che l’utente abbia selezionato lo strumento rettangolo e sia in edit mode.

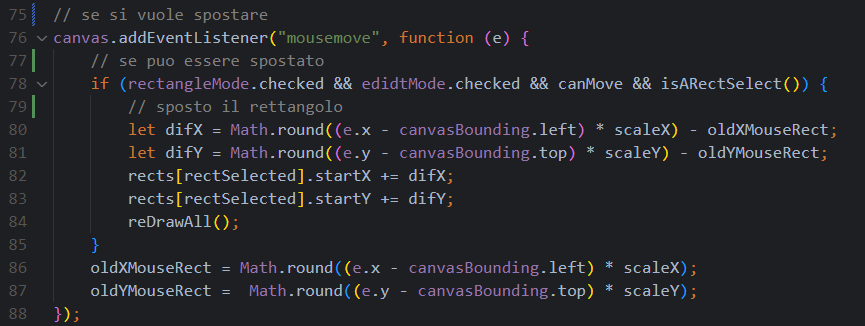
Poi per selezionare un rettangolo prendo le coordinate del click e poi passo tutti i rettangoli e guardo se queste appartengono ad un rettangolo nell’array rects.

Se le coordinate appartengono ad un rettangolo allora ritorno la posizione del rettangolo nell’array sennò ritorno -1.

Poi nel caso in qui il seguente rettangolo è già selezionato lo deseleziono, sennò lo seleziono.

#### Spostare

Il principio per spostare un rettangolo è lo stesso di quello dei puntini. La differenza sta che l’utente per spostare il rettangolo non premerà nella coordinata (0,0) ma magari lo sposta dal centro del rettangolo quindi per prendere le nuove coordinate del punto (0,0) aggiungo al punto startX e starY la differenza di spostamento tra la posizione precedente e quella nuova.



#### Altre azioni

Il funzionamento delle altre azioni è presso che uguale alle azioni fatte sui puntini con la differenza che vengono fatte sull’array rects e non bisogna rinominare i rettangoli quando ne viene eliminato un essendo che non hanno un numero.

### actionCircle

In questo file c’è il codice che permette di fare delle azioni sui cerchi.

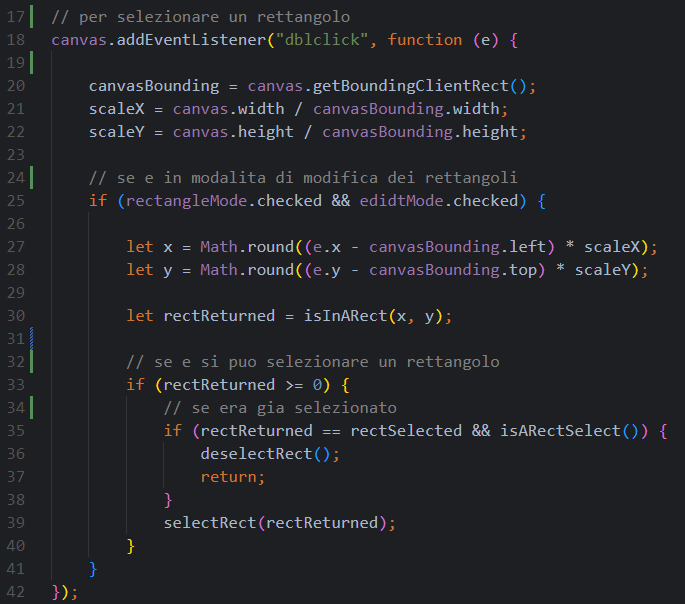
#### Selezionare

Per selezionare un cerchio ho creato una funzione anonima che per prima cosa guarda che l’utente si trova nella modalità corretta.

Poi prende le coordinate del doppio click e guarda se nell’array esiste un cerchio che contiene la seguente coordinata.

Per far questo quando guardo se la coordinata potrebbe appartenere al cerchio guardo se la distanza della coordinata dal centro è inferiore al raggio.

Poi nel caso esiste un cerchio ritorno la posizione di esso nell’array.



#### Altre azioni

Il funzionamento delle altre azioni è uguale a quello dei rettangoli e dei puntini.

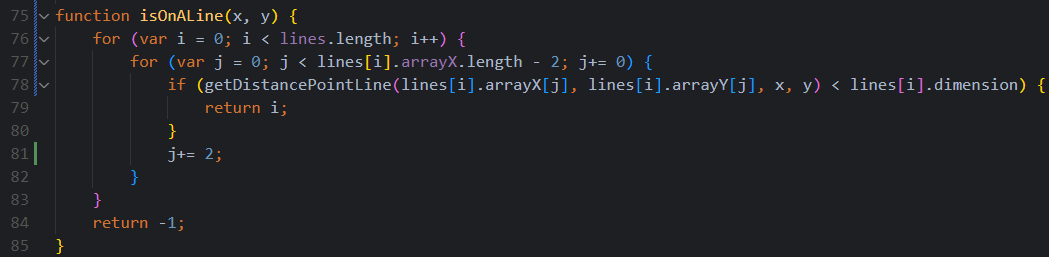
### actionLines

Questo file contiene il codice per fare le azioni sulle linee.

#### Selezionare

Per selezionare una linea ho creato una funzione anonima che viene richiamata al doppio click. Poi controllo che l’utente si trovi nella modalità corretta (strumento Pencil e edit mode attivata).

Essendo la linea salvata come un insieme di puntini per selezionarle passo tutta la line e per quasi ogni puntino controllo se il click è avvenuto su oppure da parte ad esso.

Quasi perché essendo la linea molto densa di puntini per ottimizzare il tutto passo un puntino su due per far si che ci siano meno puntini da confrontare. Questo avviene nella funzione isOnLine.



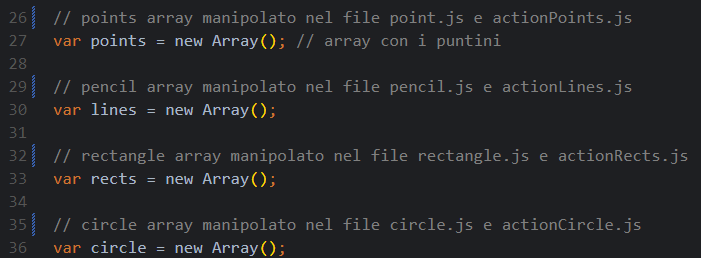
#### Altre azioni

Essendo la linea uno strumento pesante per quanto riguarda le prestazioni non do la possibilità all’utente di poter spostare le linee una volta disegnate. Ma solo di poterle cambiare di spessore, colore e cancellarle.

Il funzionamento di quest’ultime azioni è presso che identico a quanto fatto con gli altri strumenti.

## drawHelp.js

In questo file ho messo tutto quel codice che non si riferisce ad un singolo oggetto.

In cima al file ho dichiarato gli array in qui sono salvati i disegni.

Poi ho inserito i seguenti metodi: reDrawAll(), reDrawAllWhidoutClear(). Questi metodi permettono di ridisegnare tutti gli oggetti salvati negli array.

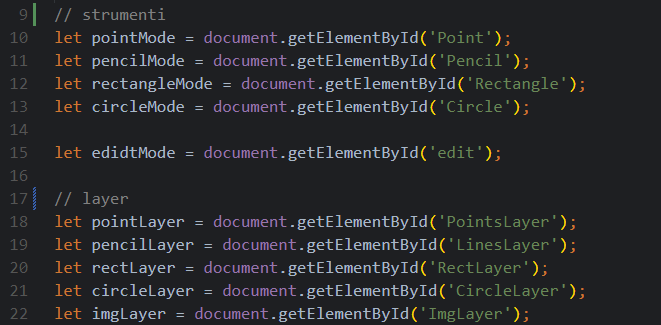
Ho inserito anche la funzione deselectAll() che consente di deselezionare tutti gli oggetti. Questo è utile per esempio prima di esportare l’immagine.

Poi ho inserito anche le funzioni connectDots() e reConnectDots(). Ho deciso di inserirli in questo file e non in actionPoints perché questi non servono per fare azioni sui puntini ma disegnano una linea che li collega.

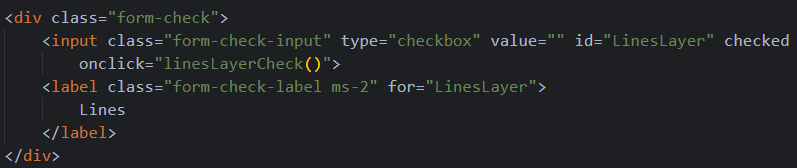
connectDots() viene richiamato solo dall’utente quando preme sul bottone apposito.

Mente reConnectDots() viene richiamato nel codice quando faccio un reDrawAll().

## main

Nel main.js ho inserito istanzio le variabili che mi permettono di controllare che opzioni l’utente ha selezionato.

Poi gestisco anche la selezione dei checkbox tramite per esempio la funzione linesLayerCheck() che controlla se il layer delle linee è visibile.





Il codice è presso che identico per ogni layer di ogni strumento.

## imageImport

In questo file c’è il codice che permette di importare l’immagine.

Tramite la funzione loadImage() quando verrà importata l’immagine verrà chiamata e farà il render dell’immagine.

Prima di caricarla nel canvas faccio un controllo che l’immagine sia abbastanza grande (per far sì che non si possa importare immagine troppo piccole), e poi la carico.

Dopo averla caricata setto le variabili per il calcolo delle coordinate e dichiaro gli array che conterranno i disegni.

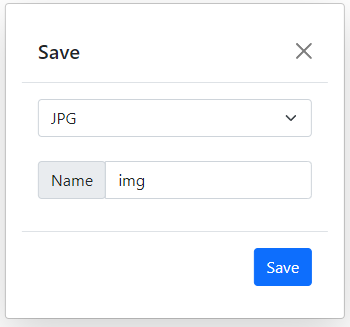
Se l’immagine sia troppo piccola verrà mostrato a schermo il seguente messaggio.

## ImageExport

Nel seguente file ho messo il codice per poter esportare l’immagine.

Prima di esportare l’immagine verrà mostrato a schermo il seguente menu per settare le impostazioni di salvataggio.

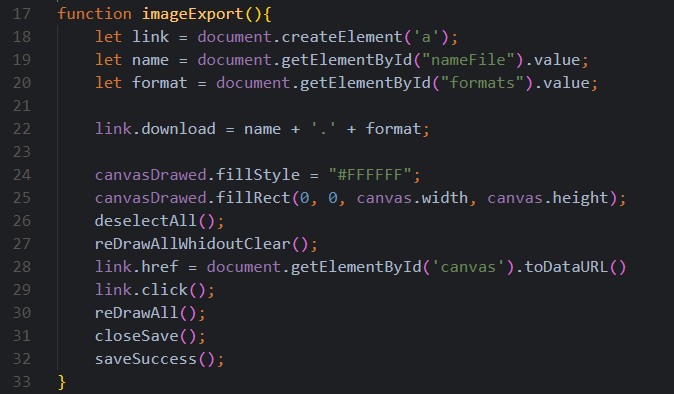
Questo viene gestito dalla funzione openSave() che apre il pannello.



Quando l’utente premerà Save verrà chiamato il metodo imageExport() che preparerà il download con il nome e il formato scelto.

Poi per far si che lo sfondo sia bianco coloro il canvas di bianco e poi ridisegno tutto senza prima pulire il canvas.

E infine la scarico. E per tornare allo stato iniziale ridisegno tutto pulendo prima il canvas per rimuovere lo sfondo bianco.



# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-01 | **Nome:** | Manipolazione puntini |
| **Descrizione:** | Disegnare, spostare, cambiare colore, ingrandire, cambiare numero, eliminare dei puntini | | |
| **Prerequisiti:** | Aver importato in immagine | | |
| **Procedura:** | 1. Creazione di un puntino 2. Selezionare il puntino 3. Ingrandirlo 4. Cambiargli il colore 5. Cambiargli il numero 6. Spostarlo 7. Eliminarlo | | |
| **Risultati attesi:** | Il puntino viene spostato, selezionato, ingrandito, viene cambiato il numero e infine eliminato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-02 | **Nome:** | Strumenti di disegno |
| **Descrizione:** | Disegnare con ogni strumento | | |
| **Prerequisiti:** | Aver importato in immagine | | |
| **Procedura:** | 1. Disegnare con la linea   Eseguire tutte le operazioni possibili   1. Disegnare con il cerchio   Eseguire tutte le operazioni possibili   1. Disegnare con il rettangolo   Eseguire tutte le operazioni possibili   1. Eliminare tutti i disegni | | |
| **Risultati attesi:** | Tutti gli strumenti vengono disegnati e modificato per quanto concesso e poi vengono eliminati. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-03 | **Nome:** | Esportare l’immagine |
| **Descrizione:** | Esportazione dell’immagine | | |
| **Prerequisiti:** | Aver importato in immagine | | |
| **Procedura:** | 1. Importare l’immagine 2. Fare un disegno 3. Esportarla con tutti i formati disponibili | | |
| **Risultati attesi:** | L’immagine viene esportata nel formato desiderato con sfondo bianco e sopra i disegni. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-04 | **Nome:** | Soluzione |
| **Descrizione:** | Deve poter essere visibile la soluzione | | |
| **Prerequisiti:** | Aver importato in immagine | | |
| **Procedura:** | 1. Creare un disegno 2. Collegare i puntini tramite l’apposito bottone 3. Rimuovere il collegamento | | |
| **Risultati attesi:** | I puntini disegnati vengono collegati da una linea retta in ordine numerico e poi ripremendo il bottone la soluzione viene cancellata. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-05 | **Nome:** | Layer |
| **Descrizione:** | Deve poter essere visibile la soluzione | | |
| **Prerequisiti:** | Aver importato in immagine | | |
| **Procedura:** | 1. Creare un disegno con tutti gli strumenti 2. Fare tutte le combinazioni possibili con i layer | | |
| **Risultati attesi:** | Durante le combinazioni vengono mostrati solo i layer selezionati. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006 | **Nome:** | Delete All |
| **Descrizione:** | Eliminare tutti i disegni in un solo colpo. | | |
| **Prerequisiti:** | Aver importato in immagine | | |
| **Procedura:** | 1. Creare un disegno con tutti gli strumenti 2. Eliminare tutto tramite Delete All | | |
| **Risultati attesi:** | Richiesta di conferma, eliminazione di tutti i disegni. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-07 | **Nome:** | Importare con più formati |
| **Descrizione:** | Poter importare immagini con più formati e dimensioni. | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Importare immagini con i seguenti formati: jpg, jpeg, png, webp | | |
| **Risultati attesi:** | L’immagine dovrà essere importata con successo. | | |

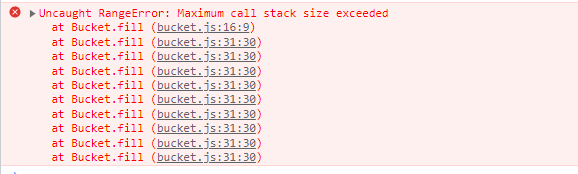
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008 | **Nome:** | Immagine importata troppo piccola. |
| **Descrizione:** | Se l’immagine importata è troppo piccola verrà mostrato un messaggio di errore. | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Importare un’immagine con dimensioni inferiori a 250x150 | | |
| **Risultati attesi:** | Verrà mostrato un messaggio di errore. | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Nome | Risultato | Problema |
| TC-001 | Manipolazione puntini | 1 |  |
| TC-002 | Strumenti di disegno | 0.8 | Strumento secchiello non fatto |
| TC-003 | Esportare l’immagine | 1 |  |
| TC-004 | Soluzione | 1 |  |
| TC-005 | Layer | 1 |  |
| TC-006 | Delete All | 1 |  |
| TC-007 | Importare con più formati | 1 |  |
| TC-008 | Immagine importata troppo piccola. | 1 |  |

## Mancanze/limitazioni conosciute

### Strumento secchiello

Durante lo sviluppo dello strumento secchiello con l’algoritmo da me sviluppato siccome usavo la ricorsione il programma usciva dallo stack. Poi per mancanza di tempo sono dovuto andare avanti.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

In futuro si potrebbe ricreare lo strumento secchiello. E ottimizzare maggiormente le linee per permettere di spostarle.

Rendere completamente responsive l’interfaccia grafica.

## Considerazioni personali

Ho imparato a gestire un progetto di dimensioni maggiori rispetto agli altri esercizi fatti in precedenza.

Le basi di git e GitHub.

Ad utilizzare meglio il canvas e Bootstrap.

Ho imparato l’importanza delle documentazioni e di non farla solo alla fine ma farla man mano col lo sviluppo del progetto.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| Array | Elemento che contiene un insieme di variabili. |
| Bootstrap | Libreria per il CSS. |
| Canvas | Elemento JavaScript che permette di disegnare facilmente. |
| CSS | Cascading Style Sheets: linguaggio che permette di definire il layout e la grafica di una pagina web. |
| Funzione / metodo | Blocco di codice che può avere parametri in entrata e può ritornare qual cosa. |

# Allegati

Elenco degli allegati, esempio:

* Gant
* Use Case