

## 1. Introduzione

- Informazioni sul progetto
- Abstract
- Scopo

## 2. Analisi

- Analisi e specifica dei requisiti
- Use case
- Pianificazione
- Analisi dei mezzi
  - Hardware
  - Software

## 3. Progettazione

- Design delle interfacce
- Design procedurale

## 4. Implementazione

## 5. Test

- Protocollo di test
- Risultati test

## 6. Consuntivo

## 7. Conclusioni

- Sviluppi futuri
- Considerazioni personali

## 8. Sitografia

## 9. Allegati

# Introduzione

---

## Informazioni sul progetto

Scuola e classe: SAMT I3AC

Modulo: 306

Alievo: Jonathan Mueller

Docenti: Luca Muggiasca e Geo Peduzzi

Durata progetto: 6.9.2019 - 20.12.2019

## Abstract

E' una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all'utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

- **Background/Situazione iniziale**
- **Descrizione del problema e motivazione:** Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
- **Approccio/Metodi:** Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche...)? Quale è stata l'entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
- **Risultati:** Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di abstract:

*As the size and complexity of today's most modern computer chips increase, new techniques must be developed to effectively design and create Very Large Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.*

## Scopo

Lo scopo di questo progetto è di imparare a gestire progetti a partire da una consegna di un cliente seguendo i modelli imparati in classe (cascata), a documentarli e a fare i diari settimanali. Durante la creazione del progetto ci saranno utili molte nozioni imparate da moduli degli anni passati, come 431, 307, 226 e il corrente 306. Questo progetto ci aiuterà anche ad allenarci per fare progetti futuri come il progetto finale in quarta e quelli che svolgeremo sul posto di lavoro.

## Analisi

### Analisi del dominio

Prima dell'invenzione di questo programma per generare un fiocco di neve si doveva disegnare a mano o ritagliare un foglio di carta. Questo prodotto è originale, e non esistevano prodotti simili prima. Non occorrono competenze per far funzionare il programma i quanto il metodo di utilizzo è descritto sulla pagina web del prodotto.

## Analisi e specifica dei requisiti

- L'applicativo può essere sia Java che in Javascript.
  - Se l'applicazione è Java, va creato un sito web con la descrizione e il download dell'applicazione, con descrizione di tutti i requisiti per il funzionamento.
  - Se l'applicazione è in Javascript va creato un sito web per il suo hosting.
- I punti di "taglio" vengono inseriti cliccando col mouse.
- I punti si devono poter aggiungere e resettare completamente.
  - Bonus: rimozione, spostamento di punti
- Il "fiocco di neve" viene generato quando l'utente clicca il tasto "Genera"
  - Bonus: la generazione avviene in tempo reale con una animazione
- Si può salvare il fiocco di neve come immagine raster in formato PNG e vettoriale in formato SVG (l'utente decide le dimensioni).
- L'applicativo deve permettere di salvare i punti di taglio per poter permettere modifiche
  - rigenerazione future.
- In base al tempo a disposizione, nuovi requisiti possono essere inseriti nel progetto dopo discussione fra formatore e allievo.

ID	Req-1
	Nome
	Priorità
	Versione
	Note
	001
	002

ID	Req-2
	Nome
	Priorità

	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-3
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-4
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-5
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-6
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>

	<b>Note</b>
--	-------------

<b>ID</b>	Req-7
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-8
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-9
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-10
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>

**Note**

<b>ID</b>	<b>Req-11</b>
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>
	001

<b>ID</b>	<b>Req-12</b>
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	<b>Req-13</b>
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	<b>Req-14</b>
	<b>Nome</b>
	<b>Priorità</b>

	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-15
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>

<b>ID</b>	Req-16
	<b>Nome</b>
	<b>Priorità</b>
	<b>Versione</b>
	<b>Note</b>
	001
	002

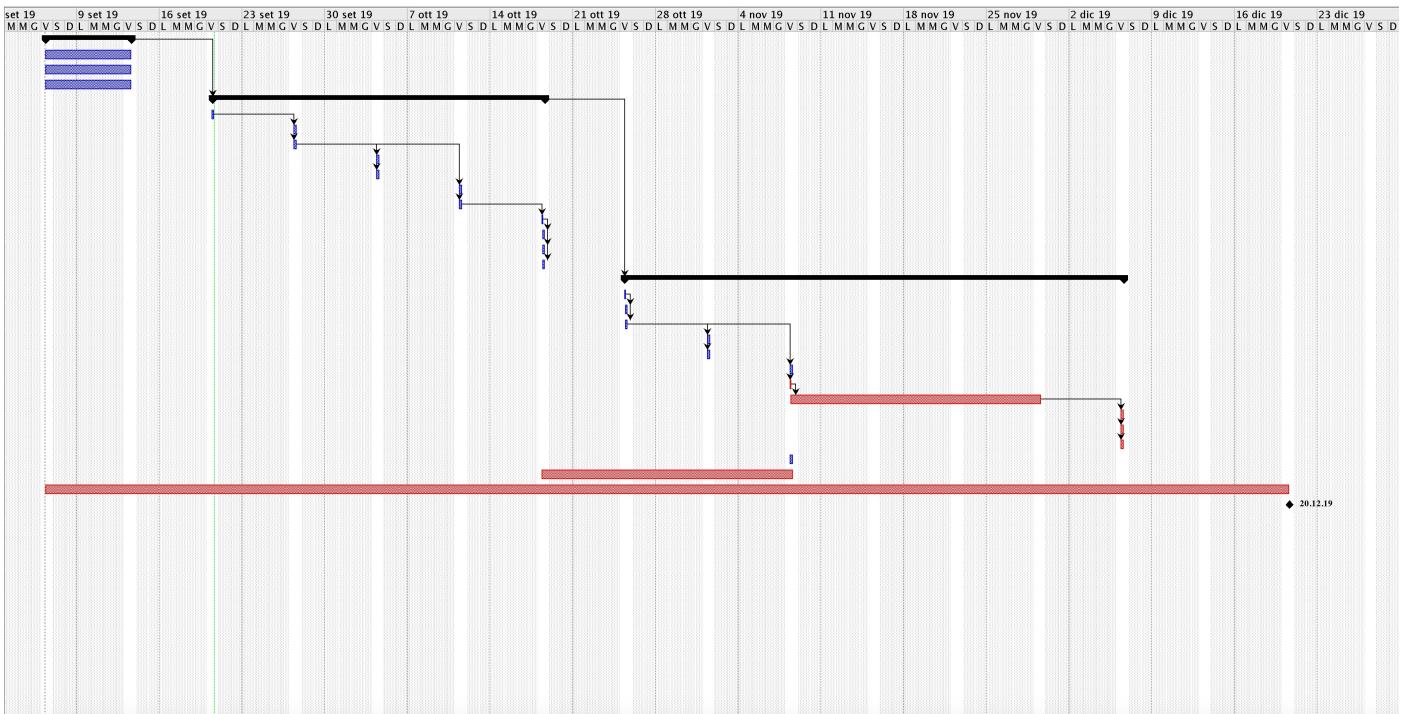
## Use case

È presente solo un'attore che interagisce direttamente con l'applicazione.

## Pianificazione

	<b>Nome</b>	<b>Durata</b>	<b>Avvio</b>	<b>Termine</b>
1	<input checked="" type="checkbox"/> Requisiti	1.188 gio...	06.09.19 10:05	13.09.19 16:30
2	Definizione requisiti	1.188 gio...	06.09.19 10:05	13.09.19 16:30
3	Discussione requisiti	1.188 gio...	06.09.19 10:05	13.09.19 16:30
4	Use case	1.188 gio...	06.09.19 10:05	13.09.19 16:30
5	<input checked="" type="checkbox"/> Progettazione	2.781 gio...	20.09.19 13:15	18.10.19 16:30





## Analisi dei mezzi

### Hardware

PC messo a disposizione dalla scuola con Windows

Portatile MacBook Pro personale

### Software

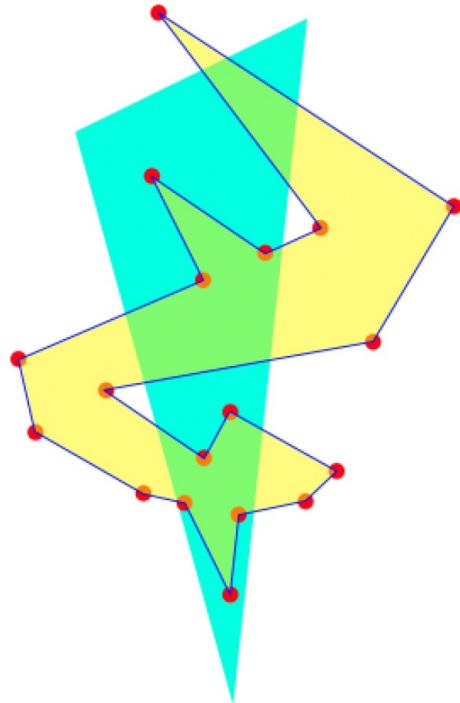
Java (JDK 11.1)

GitHub

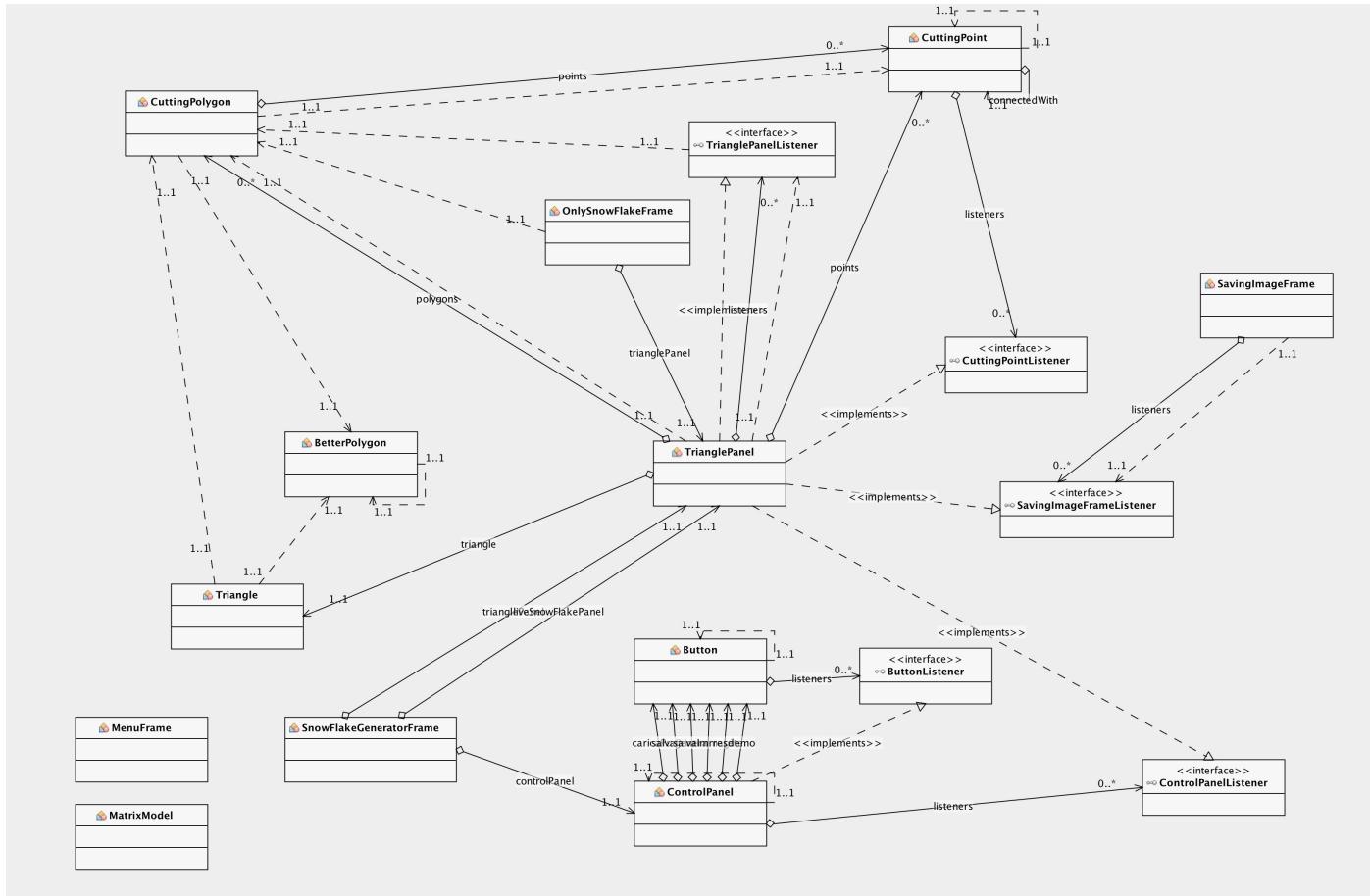
## Progettazione

### Design delle interfacce

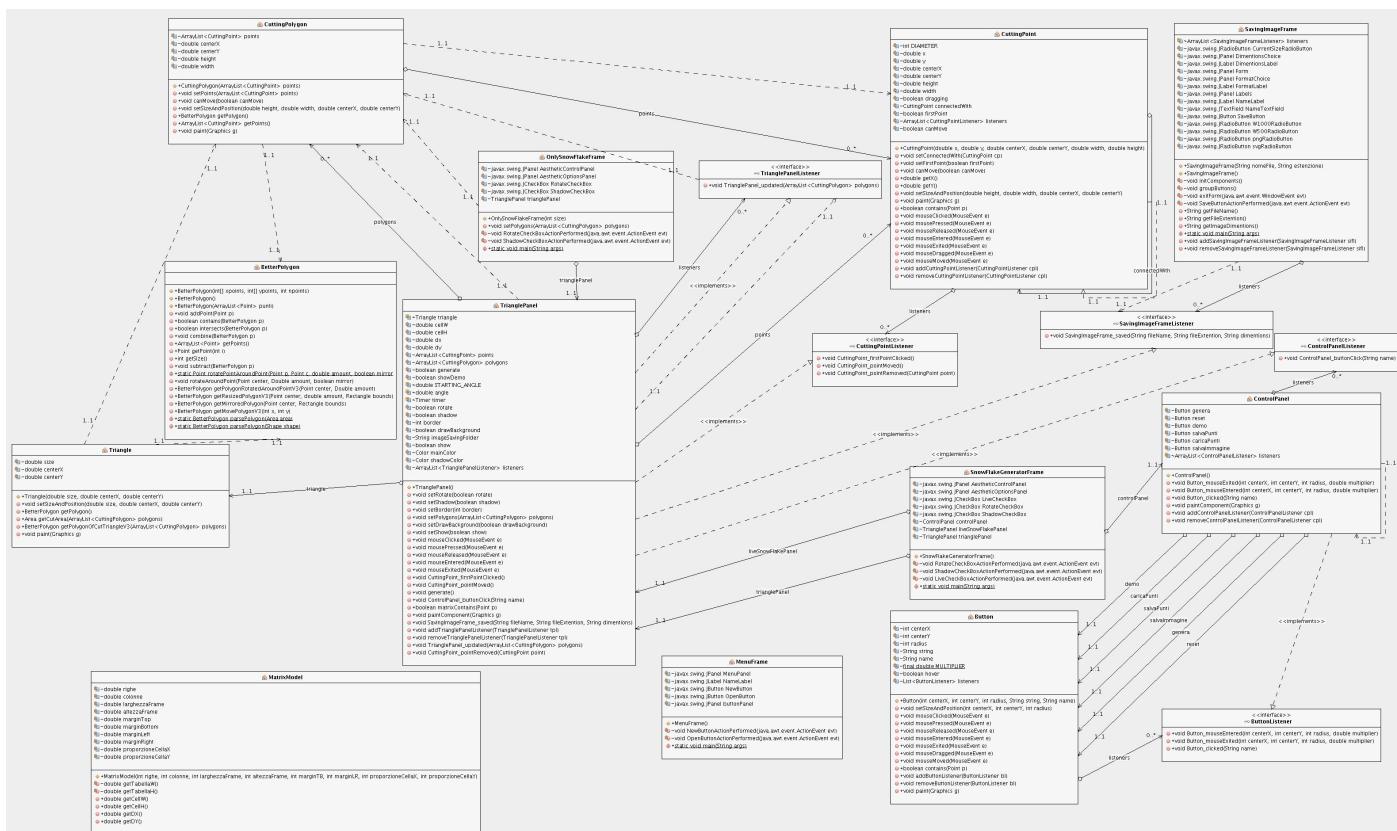
Menu
Save
Load
Generate



## Design procedurale



# Implementazione



## Metodi complicati

Questi sono stati i metodi più complicati e che mi hanno dato più problemi:

Conversione da Area a BetterPolygon

```
/*
 * Trasforma un'area in un poligono.
 * @param area L'area da trasformare
 * @return Il poligono
 */
public static BetterPolygon parsePolygon(Area area){
    PathIterator iterator = area.getPathIterator(null);
    float[] floats = new float[6];
    BetterPolygon polygon = new BetterPolygon();
    Point closingPoint = null;
    ArrayList<Point> closingPoints = new ArrayList<>();
    while (!iterator.isDone()) {
        int type = iterator.currentSegment(floats);
        int px = (int) floats[0];
        int py = (int) floats[1];
        polygon.addPoint(px, py);
        if(type == 0){
            closingPoint = new Point(px,py);
            closingPoints.add(closingPoint);
        }
        if(type == 4){
            polygon.addPoint(closingPoint);
        }
        iterator.next();
    }
    for (Point finalClosingPoint : closingPoints) {
        polygon.addPoint(finalClosingPoint);
    }
    return polygon;
}
```

```

/**
 * Trasforma una shape in un poligono.
 * @param shape La shape da trasformare
 * @return Il poligono
 */
public static BetterPolygon parsePolygon(Shape shape){
    PathIterator iterator = shape.getPathIterator(null);
    float[] floats = new float[6];
    BetterPolygon polygon = new BetterPolygon();
    Point closingPoint = null;
    ArrayList<Point> closingPoints = new ArrayList<>();
    while (!iterator.isDone()) {
        int type = iterator.currentSegment(floats);
        int px = (int) floats[0];
        int py = (int) floats[1];
        polygon.addPoint(px, py);
        if(type == 0){
            closingPoint = new Point(px,py);
            closingPoints.add(closingPoint);
        }
        if(type == 4){
            polygon.addPoint(closingPoint);
        }
        iterator.next();
    }
    for (Point finalClosingPoint : closingPoints) {
        polygon.addPoint(finalClosingPoint);
    }
    return polygon;
}

```

Arrivato al punto della sottrazione dei poligoni al triangolo mi sono ritrovato con un area quando mi serviva un poligono per fare tutte le operazioni successive. Per convertire ho usato il PathIterator, che passa per tutti i punti dell'area. Questi punti possono avere diversi tipi: 0 Inizio di un poligono 1 Punto di mezzo 4 Fine del poligono Sapendo questo ho aggiunto i punti ad una array in modo da avere tutti i poligoni salvati come uno solo. Alla fine ho creato un poligono con questi punti.

Ridimensionamento, specchiamento e rotazione dei poligoni

```

/**
 * Ritorna una copia di questo poligono roteato attorno ad un punto.
 * @param center Il punto da ruotare attorno
 * @param amount Di quanto ruotare in gradi
 * @return Il poligono roteato
 */
public BetterPolygon getPolygonRotatedAroundPointV3(Point center, Double amount){
    AffineTransform at = new AffineTransform();
    at.rotate(Math.toRadians(amount), center.x, center.y);
    Shape shape = at.createTransformedShape(this);
    BetterPolygon polygon = BetterPolygon.parsePolygon(shape);
    return polygon;
}

/**
 * Ritorna una copia del poligono ridimensionato.
 * @param center Il centro che rimarrà fermo
 * @param amount Le proporzioni del ridimensionamento
 * @param bounds Lo spazio che il poligono occupava (nel caso sia stato tagliato)
 * @return Il poligono ridimensionato
 */
public BetterPolygon getResizedPolygonV3(Point center, double amount, Rectangle bounds){
    AffineTransform at = new AffineTransform();
    int x = bounds.x;
    int y = bounds.y;
    at.translate(-x, -y);
    at.scale(amount, amount);
    at.translate(x*3, y*3);
    Shape shape = at.createTransformedShape(this);
    BetterPolygon polygon = BetterPolygon.parsePolygon(shape);
    return polygon;
}

/**
 * Ritorna una copia del poligono specchiato da un asse.
 * @param center Un punto dell'asse
 * @param bounds Lo spazio che il poligono occupava (nel caso sia stato tagliato)
 * @return Il poligono specchiato
 */
public BetterPolygon getMirroredPolygon(Point center, Rectangle bounds){
    AffineTransform at = new AffineTransform();
    int x = bounds.x;
    int y = bounds.y;
    at.translate(-x, -y);
    at.scale(-1, 1);
    at.translate(-(x * 2 + bounds.width + center.x), y);
    Shape shape = at.createTransformedShape(this);
    BetterPolygon polygon = BetterPolygon.parsePolygon(shape);
    return polygon;
}

```

Inizialmente avevo fatto queste tre operazioni a mano, modificando i poligoni punto per punto, ma questa operazione era molto pesante e causava lag quando c'erano molti punti. Quindi ho cercato un altro modo e ho trovato `AffineTransform`. Questa classe permette di fare molte operazioni su shapes tra cui quelle elencate sopra, con una assenza quasi totale di lag. Il problema è che questa classe è estremamente complicata da usare e ci ho messo un

po' a capire come usarla.

## 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	TC-001
<b>Nome</b>	I punti di taglio si posizionano con il click del mouse
<b>Riferimento</b>	REQ-04
<b>Descrizione</b>	I punti di taglio si posizionano con il click sinistro del mouse sull'area di lavoro
<b>Prerequisiti</b>	Applicazione già avviata sull'editor
<b>Procedura</b>	- Cliccare nell'area di lavoro con il tasto destro
<b>Risultati attesi</b>	Viene generato un punto sull'area di lavoro

Test Case	TC-002
<b>Nome</b>	Mettere almeno 3 punti e cliccare il punto verde crea un poligono
<b>Riferimento</b>	REQ-04
<b>Descrizione</b>	Quando ho posizionato almeno 3 punti e clicco sul punto verde si crea un poligono
<b>Prerequisiti</b>	Applicazione già avviata sull'editor
<b>Procedura</b>	- Cliccare nell'area di lavoro con il tasto destro per posizionare più punti - Una volta posizionati almeno 3 punti cliccare il punto verde
<b>Risultati attesi</b>	Viene generato un poligono sull'area di lavoro

Test Case	TC-003
<b>Nome</b>	I punti di taglio si posizionano resettare

<b>Riferimento</b>	REQ-06
<b>Descrizione</b>	I punti di taglio si possono resettare cliccando il tasto "reset"
<b>Prerequisiti</b>	Applicazione già avviata sull'editor e punti posizionati
<b>Procedura</b>	- Cliccare il tasto "reset"
<b>Risultati attesi</b>	Tutti i punti spariscano

<b>Test Case</b>	TC-004
<b>Nome</b>	I punti di taglio si possono eliminare
<b>Riferimento</b>	REQ-07
<b>Descrizione</b>	I punti di taglio si possono eliminare cliccando con il tasto destro
<b>Prerequisiti</b>	Applicazione già avviata sull'editor e punti posizionati
<b>Procedura</b>	- Cliccare con il tasto destro su un punto
<b>Risultati attesi</b>	Il punto cliccato viene eliminato

<b>Test Case</b>	TC-005
<b>Nome</b>	I punti di taglio si possono spostare
<b>Riferimento</b>	REQ-08
<b>Descrizione</b>	I punti di taglio si possono spostare trascinandoli con il mouse
<b>Prerequisiti</b>	Applicazione già avviata sull'editor e punti posizionati
<b>Procedura</b>	- Trascinare il punto con il mouse
<b>Risultati attesi</b>	Il punto segue il mouse

<b>Test Case</b>	TC-006
<b>Nome</b>	Il fiocco di neve viene generato al click del pulsante "Genera"

<b>Riferimento</b>	REQ-08
<b>Descrizione</b>	Quando clicco il tasto "Genera" il triangolo viene tagliato e il fiocco viene generato
<b>Prerequisiti</b>	Applicazione già avviata sull'editor e punti posizionati
<b>Procedura</b>	- Cliccare il tasto genera
<b>Risultati attesi</b>	Il triangolo tagliato viene moltiplicato 12 volte in modo da formare il fiocco

<b>Test Case</b>	TC-007
<b>Nome</b>	Il fiocco di neve viene generato in tempo reale
<b>Riferimento</b>	REQ-09
<b>Descrizione</b>	Quando posiziono un nuovo punto la live viene aggiornata. Si può anche disattivare la generazione in tempo reale
<b>Prerequisiti</b>	Applicazione già avviata sull'editor
<b>Procedura</b>	- Posiziono dei punti sull'area di lavoro - Clicco la checkbox per disattivare la live
<b>Risultati attesi</b>	Il fiocco in live viene aggiornato, e scompare se la checkbox viene deselectata

<b>Test Case</b>	TC-008
<b>Nome</b>	Si può salvare il fiocco come immagine PNG o SVG a scelta con dimensioni scelte dall'utente (500px o 1000px)
<b>Riferimento</b>	REQ-12,REQ-13,REQ-14
<b>Descrizione</b>	Cliccando il tasto "Esporta" si può salvare il fiocco come immagine
<b>Prerequisiti</b>	Applicazione già avviata sull'editor e poligoni posizionati
<b>Procedura</b>	- Clicco il tasto "Esporta" - Nel file chooser seleziono un file ".png" o ".svg" in cui salvare - Clicco "Save" - Nel menu seleziono i dettagli del salvataggio - Clicco "Salva"
<b>Risultati attesi</b>	Nel file selezionato verrà salvata l'immagine. Se il file non esiste viene creato

Test Case	TC-009
<b>Nome</b>	Si possono salvare i punti di taglio per modifiche future
<b>Riferimento</b>	REQ-15
<b>Descrizione</b>	Cliccando il tasto "Salva" si possono salvare i punti in un file
<b>Prerequisiti</b>	Applicazione già avviata sull'editor e punti posizionati
<b>Procedura</b>	- Clicco il tasto "Salva" - Nel file chooser seleziono un file ".points" in cui salvare - Clicco "Save"
<b>Risultati attesi</b>	Nel file selezionato verranno salvati i punti. Se il file non esiste viene creato

Test Case	TC-010
<b>Nome</b>	Si possono caricare i punti di taglio salvati in precedenza
<b>Riferimento</b>	REQ-16
<b>Descrizione</b>	Cliccando il tasto "Carica" si possono caricare i punti da un file
<b>Prerequisiti</b>	Applicazione già avviata sull'editor e file di punti già creato
<b>Procedura</b>	- Clicco il tasto "Carica" - Nel file chooser seleziono un file ".points" da cui caricare - Clicco "Open"
<b>Risultati attesi</b>	I punti e i poligoni vengono creati sul piano di lavoro

## Risultati test

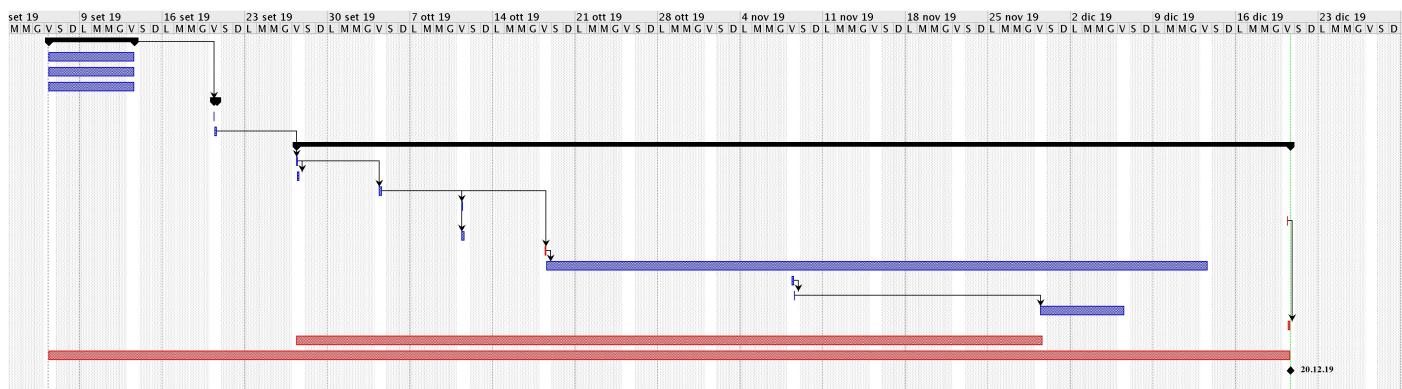
Test Case	Risultato
TC-001	Il punto viene posizionato
TC-002	Il poligono viene generato

TC-003	Tutti i punti vengono cancellati
TC-004	Il punto viene eliminato. Certe volte questo non funziona, e la hitbox dei punti è sbagliata. Se questo succede si può chiudere l'editor e aprirne uno nuovo, ma non sempre questo risolve il problema
TC-005	Il punto segue il mouse. Come nel TC-004 succede lo stesso problema, e lo si può provare a risolvere nello stesso modo
TC-006	Il fiocco di neve viene generato
TC-007	Il fiocco di neve viene generato in tempo reale. Se si disattiva la preview scompare
TC-008	Le immagini vengono salvate, e nel caso il file non esiste viene creato
TC-009	I punti vengono salvati solo se seleziono un file ".points", e nel caso il file non esiste viene creato

## Consuntivo

La creazione del fiocco di neve è durata molto più del previsto

	Nome	Durata	Avvio	Termine	
1	▣ Requisiti	1.188 giorni	06.09.19 10:05	13.09.19 16:30	
2	Definizione requisiti	1.188 giorni	06.09.19 10:05	13.09.19 16:30	
3	Discussione requisiti	1.188 giorni	06.09.19 10:05	13.09.19 16:30	
4	Use case	1.188 giorni	06.09.19 10:05	13.09.19 16:30	
5	▣ Progettazione	0.594 giorni	20.09.19 10:05	20.09.19 16:30	1
6	Gestione interfaccia	0.187 giorni	20.09.19 10:05	20.09.19 11:34	
7	Gantt	0.406 giorni	20.09.19 13:15	20.09.19 16:30	
8	▣ Implementazione	7.719 giorni	27.09.19 10:05	20.12.19 16:30	
9	Gestione interfaccia	0.188 giorni	27.09.19 10:05	27.09.19 13:15	7
10	Gestione triangolo	0.406 giorni	27.09.19 13:15	27.09.19 16:30	9
11	Gestione posizionamento	0.594 giorni	04.10.19 10:05	04.10.19 16:30	9
12	Gestione reset punti	0.188 giorni	11.10.19 10:05	11.10.19 13:15	11
13	Gestione eliminazione	0.062 giorni	20.12.19 10:05	20.12.19 10:35	
14	Gestione spostamento	0.594 giorni	11.10.19 10:05	11.10.19 16:30	11
15	Gestione tasto genera	0.188 giorni	18.10.19 13:15	18.10.19 14:45	11
16	Gestione generazione	4.969 giorni	18.10.19 14:45	13.12.19 16:30	15
17	Gestione salvataggio file	0.312 giorni	08.11.19 10:05	08.11.19 14:15	
18	Gestione caricamento	0.281 giorni	08.11.19 14:15	08.11.19 16:30	17
19	Gestione salvataggio immagine	0.812 giorni	29.11.19 11:05	06.12.19 14:30	18
20	Creazione sito web	0.531 giorni	20.12.19 10:35	20.12.19 16:30	13
21	Test	5.938 giorni	27.09.19 10:05	29.11.19 16:30	
22	Documentazione	9.5 giorni?	06.09.19 10:05	20.12.19 16:30	
23	Consegna progetto	0 giorni?	20.12.19 16:30	20.12.19 16:30	



## Conclusioni

L'utente potrà finalmente creare un fiocco di neve e non dovrà più farlo a mano.

## Sviluppi futuri

Il programma è completo e, almeno che il cliente non chiede di aggiungere qualcosa di nuovo, non ci saranno sviluppi futuri.

## Considerazioni personali

Da questo progetto ho migliorato la mia capacità di programmatore e ho imparato ad usare nuove class come `AffineTransform`.

## Bibliografia

---

### Sitografia

- <https://github.com>, *github*, 20-12-2019.
- <https://stackoverflow.com>, *stackoverflow*, 20-12-2019.
- <https://docs.oracle.com/javase/7/docs/api/>, *javadoc*, 20-12-2019

## Allegati

---

### Diari

- [Diario 1](#)
- [Diario 2](#)
- [Diario 3](#)
- [Diario 4](#)
- [Diario 5](#)
- [Diario 6](#)
- [Diario 7](#)
- [Diario 8](#)
- [Diario 9](#)
- [Diario 10](#)
- [Diario 11](#)
- [Diario 12](#)
- [Diario 13](#)
- [Diario 14](#)
- [Diario 15](#)