Contents

[Introduzione 2](#_Toc39649015)

[Informazioni sul progetto 2](#_Toc39649016)

[Abstract 2](#_Toc39649017)

[Scopo 2](#_Toc39649018)

[Analisi 2](#_Toc39649019)

[Analisi del dominio 2](#_Toc39649020)

[Use case 4](#_Toc39649021)

[Pianificazione 4](#_Toc39649022)

[Analisi dei mezzi 4](#_Toc39649023)

[Progettazione 5](#_Toc39649024)

[Design dell’architettura del sistema 5](#_Toc39649025)

[Implementazione 6](#_Toc39649026)

[Elenco delle classi 7](#_Toc39649027)

[Ridimensionamento del triangolo e del poligono 7](#_Toc39649028)

[Aggiunta e rimozione dei punti 10](#_Toc39649029)

[Salvataggio e importazione dei punti 13](#_Toc39649030)

[Creazione del fiocco di neve 13](#_Toc39649031)

[Cambiamento di design 14](#_Toc39649032)

[Test 15](#_Toc39649033)

[Protocollo di test 15](#_Toc39649034)

[Risultati test 18](#_Toc39649035)

[Mancanze/limitazioni conosciute 18](#_Toc39649036)

[Consuntivo 19](#_Toc39649037)

[Conclusioni 19](#_Toc39649038)

[Sviluppi futuri 19](#_Toc39649039)

[Considerazioni personali 19](#_Toc39649040)

# Introduzione

## Informazioni sul progetto

* **Allievi coinvolti (Progettisti e Sviluppatori):** 
  + Matteo Arena
  + Bruno Gomes
  + Luca Mazza
  + Jonas Bertossa
* **Docenti coinvolti (Consulenti e Clienti):** Geo Petrini e Luca Muggiasca
* **Scuola:** Scuola Arti e Mestieri (SAM) Trevano, sezione informatica
* **Classe:** I3AA.
* **Materia:** Modulo 306.
* **Durata progetto:** 17.01.2020 - 08.05.2020

## Abstract

Con questo progetto si intende realizzare un programma che permetta di bloccare il PC quando il utente non si trova davanti alla webcam.

## Scopo

Questo progetto ci offre l'opportunità di provare per la prima volta a gestire un progetto in team. I punti importanti, oltre al programma effettivo, sono la documentazione e i diari che devono essere completi ed esaustivi.

# Analisi

## Analisi del dominio

Il progetto è realizzato in maniera che possa essere utilizzato da qualsiasi utente, dal più esperto a quello che usa il computer solo per navigare online. Partendo con questo intento abbiamo decioso di rendere il tutto User-Friendly e il più comprensibile possibile in modo da non far emergere ambiguità. Cercando online non abbiamo trovato alcun programma che si occupi di questo (o almeno nessuno fatto rigorosamente bene) e, in quelli che abbiamo trovato, richiedevano tutti una conoscenza tecnica abbastanza elevata.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Nome** | **Descrizione** | **Importanza** | **Versione** |
| 0 | Riconoscere Volto | Capire quando qualcuno si trova davanti alla webcam | Alta | 1.0 |
| 1 | Registrare utente | L'utente deve poter registrare il proprio volto | Alta | 1.0 |
| 2 | Associare volto all'utente | Fare in modo che in base al volto visto dalla webcam il software capisca di che utente si tratta | Alta | 1.0 |
| 3 | Intervallo di blocco | Dopo x tempo che non è presente il volto il PC si deve bloccare | Alta | 1.0 |
| 4 | Interfaccia grafica | Tutte le impostazioni devono essere modificate da un'interfaccia grafica (electron o java) desktop | Alta | 1.0 |
| 5 | Bloccare PC | Il PC si deve bloccare quando non viene riconosciuto un volto | Alta | 1.0 |
| 6 | Scelta intervallo di blocco | L'utente può scegliere dopo quanto tempo il PC si deve bloccare | Media | 1.0 |
| 7 | Modifica impostazioni in modo sicuro | La modifica delle impostazioni richiede una password | Alta | 1.0 |
| 8 | Aggiornamento progressivo volto | I volti vengono sovrascritti ogni giorno per fare in modo che sia il più duraturo possibile | Bassa | 1.0 |
| 9 | Registrazione di più volti | Più utenti (sulla macchina) possono aggiungere il proprio volto | Media | 1.0 |
| 10 | Utente corrente | L'utente da riconoscere deve essere quello dell'account | Bassa | 1.0 |
| 11 | Multipiattaformità | Il software deve poter essere eseguito su i sistemi operativi più usati (Windows, macOs, Linux) | Alta | 1.0 |
| 12 | Cifratura dati volto | I dati del volto sono cifrati | Bassa | 1.0 |
| 13 | Countdown | Quando non viene riconosciuto un volto parte il countdown e poi si spegne | Alta | 1.0 |
| 14 | Shortcut | Tramite una shortcut deve essere possibile attivare e disabilitare le funzionalità del software | Bassa | 1.0 |
| 15 | Utenti aggiuntivi | Dentro ad un account ci possono essere anche degli utenti con il permesso di usare il PC | Media | 1.0 |
| 16 | Mail/notifica di blocco | Quando il PC viene bloccato ci arriverà una mail/notifica | Bassa | 1.0 |
| 17 | Mail/Notifica utilizzo | Quando qualcuno usa il PC e non sei tu arriva una notifica/mail | Bassa | 1.0 |

## Use case

<img src="Img/useCase.png" alt="Immagine ritraente lo use case"/>

## Pianificazione

### Pianificazione iniziale

<img src="img/GanttIniziale1.png" alt="Gantt iniziale 1"/>

<img src="img/GanttIniziale2.png" alt="Gantt iniziale 2"/>

### Pianificazione finale

<img src="img/GanttFinale1.png" alt="Gantt Finale 1"/>

<img src="img/GanttFinale2.png" alt="Gantt Finale 2"/>

Come si può notare facilmente la durata del progetto è stata estesa, questo allungamento non è stato a causa della difficoltà sottovalutata ma perché abbiamo deciso di aggiungere degli elementi aggiuntivi. Il 5 febbraio infatti avevamo già finito la grande maggioranza dei requisiti ma per rendere il rpgoramma più completo abbiamo aggiunto ancora altri requisiti.

## Analisi dei mezzi

### Hardware

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Nome** | **Proprietario** | **Sistema operativo** | **RAM** | **Storage** | **Scheda grafica** | **CPU** | **Risoluzione** |
| Acer Aspire 5 | Matteo | Windows 10 1909 | 16GB | 128GB SSD 1TB HDD | NVIDIA 940mx | i7 7500U | 1920x1080 |
| MacBook Pro (15-inch, 2017) | Luca | macOS Catalina 10.15.4 | 16GB | 256GB SSD | Radeon Pro 555 | i7 7700HQ | 2880x1800 |
| Acer Predator Orion 3000 | Matteo | Windows 10 1903 | 16GB | 500GB SSD 1TB HDD | NVIDIA Geforce RTX 2060 super | i7 9700 | 1920x1080 (monitor esterno) |
| Asus | Bruno | Windows 10 1909 | 4GB | 128GB | - | i3 7020U | 1366x768 |

### Software

|  |  |
| --- | --- |
| Nome | Versione |
| Python | 3.6.6 |
| cmake | 3.16.3 |
| dlib | 19.7.0 |
| face-recognition | 1.3.0 |
| face-recognition-models | 0.3.0 |
| numpy | 1.18.2 |
| opencv-python | 4.1.0.25 |
| pip | 20.0.2 |
| psutil | 5.7.0 |
| netbeans | 11.1 |
| JRE | 12.0.2 |
| JDK | 12.0.2 |
| NSIS | 3.05 |

# Progettazione

## Design dell’architettura del sistema

\*\*Diagramma UML\*\*

<img src="Img/UML.svg" alt="Diagramma UML del progetto"/>

### Design delle interfacce

L'applicativo si dividerà in 4 schermate, più precisamente:

1. \*\*Pagina Iniziale\*\*

<img src="Img/HomePage.jpg" alt="Pagina iniziale">

Questa sarà la schermata visualizzata all'apertura del programma. A sinistra si possono trovare i fiocchi di neve recenti, in alto a destra il bottone per creare un nuovo fiocco di neve e in mezzo una breve descrizione del programma.

2. \*\*Schermata principale\*\*

<img src="Img/SchedaPrincipale.jpg" alt="Schermata principale">

SChermata dalla quale si potrà modificare il triangolo per poi generare il fiocco di neve. I bottoni sono posti in maniera che in alto ci siano quelli per la modifica dei punti ed in basso quelli per il salvataggio e la creazione del fiocco di neve. Questa finestra avrà una dimensione minima di 1024 x 768 pixel e il triangolo occuperà sempre il 50% della finestra. In basso a destra si potrà trovare un piccolo pannello dove, se selezioniamo l'opzione "live gen", verrà visualizzato il fiocco di neve live mentre modifichiamo i punti sul triangolo. I punti sul triangolo manterranno sempre la stessa proporzione, in modo che se si aggiunge un punto con la finestra di dimensione x quando si ridimensionerà i punti rimarranno sempre nello stesso posto senza modificare il taglio effettivo.

3. \*\*Schermata di salvataggio\*\*

<img src="Img/MenuSalvataggio.jpg" alt="Schermata di salvataggio">

Grazie a questa schermata, richiamata dal bottone "Save" e "Create Snapshot" si potranno salvare i propri dati. La struttura è fatta in modo da essere intuitiva per l'utente, proprio per questo basilare.

4. \*\*Sito Web\*\*

<img src="Img/SitoWeb.jpg" alt="Sito Web">

Il Sito web è molto basilare ma contenente tutte le informazioni necessarie, molto probabilmente cambierà siccome verrà aggiunto un tab per la guida riguardate l'applicativo.

# Implementazione

## Elenco delle classi

- <b>TriangleModel: </b> Questa classe è stata sviluppata per fare in modo che un triangolo tagliato può sempre essere ridisegnato in un altro pannello con dimensioni diverse e comunque mantenere sempre le stesse proporzioni (con anche i punti disegnati sopra). Per fare in modo che questo avvenga la classe richiederà sempre la dimensione del pannello corrente quando si interagisce con essa in modo da poter trasformare i punti aggiunti/rimossi nelle dimensioni del modello. Questa classe verrà poi richiamata da praticamente tute le altre all'interno del progetto, in questa maniera le varie classi possono condividersi il triangolo senza passarsi troppi argomenti ma semplicemente un oggetto TriangleModel.

- <b>Preview</b>: Pannello che si occupa dio mostrare due tipi di preview durante il salvataggio, più specificatamente:

1) <i>Salvataggio dei punti</i> quando verranno salvati i punti nel frame principale verrà affiancato un pannello contenente la preview del triangolo con i tagli.

2) <i>Salvataggio del fiocco</i> quiando si salva l'immagine del fiocco di neve la preview cambierà e diventerà al posto del triangolo quella del fiocco di neve. Lo spazio occupato per entrambi sarà sempre lo stesso.

- <b>TrianlgePanel</b>: Interagiscew con la classe TriangleModel, i listener per il mouse sono situati in questa classe, che poi passa i punti al TriangleModel e in caso di esportazione lo passa lui. In questa maniera la classe TriangleModel rimane estremamente pulita e facilmente riutilizzabile in altre classi completamente diverse.

- <b>SaveFrame</b>: Frame di salvataggio, sia dei punti che del fiocco di neve (lo capisce grazie ad un valore booleano). In entrambi i casi però ci sarà diviso in due parti, a sinistra la preview di quello che si vuole salvare e a destra la scelta della directory e del nome del file.

- <b>SnowFrame</b>: Frame contenente il fiocco di neve generato, la generazione avverrà tramite un metodo setter che si occuperà di prendere il triangleModel e poi lo ruoterà in modo da generarlo in maniera perfetta. Adottando questa maniera di generare il fiocco il programma si può permettere di generare il fiocco in live, ovvero mentre tagliamo il triangolo in automatico il fiocco viene rigenerato (grazie ad una interfaccia del TrianglePanel).

- <b>TutorialFrame</b>: Frame che si occupa di dare una mini guida all'interno del programma all'utente, più specificatamente di ogni bottone che si visualizza a schermo e di ogni funzione. Questa guida è un poco meno completa di quella che si può trovare sul [sito ufficiale](http://www.samtinfo.ch/i17aremat/SnowFlake/).

- <b>TriangleEditFrame</b>: Frame principale del programma, quello che parte quando si avvia. Ovviamente è anche quello che interagisce con tutte le altre classi a parte TriangleModel perché con quello ci interagiscono tutte le altri classi.

## Ridimensionamento del triangolo e del poligono

Per ridimensionare il triangolo viene utilizzata la classe TriangleModel che, passandogli semplicemente la dimensione del pannello ritorna in automatico il triangolo che occupa il 50% dello spazio disponibile e il triangolo di taglio (se c'è).

Il metodo seguente è quello utilizzato per calcolare il triangolo che l'utente dovrà poi disegnare. Come si può notare il metodo è void, questo perché il triangolo non verrà calcolato unicamente quando lo si vuole disegnare ma ogni volta che si interagisce con la classe TriangleModel. Questo permette alla classe di sapere sempre su che triangolo sta lavorando l'utente e ridimensionare da solo il poligono di taglio.

/\*\*

\* Calcola il triangolo ridimensionato in base alla grandezza della

\* finestra.

\*

\* @param windowWidth Larghezza della finestra.

\* @param windowHeight Altezza della finestra.

\*/

private void calculateTriangleFromWindowSize(int windowWidth, int windowHeight) {

double minimumWidthMargin = windowWidth \* 1 / 3;

double minimumHeightMargin = windowHeight \* 1 / 4;

double maximumWidth = windowWidth - minimumWidthMargin \* 2.0;

double maximumHeight = windowHeight - minimumHeightMargin \* 2.0;

double finalWidth = TRIANGLE\_WIDTH \* maximumHeight / TRIANGLE\_HEIGHT;

double finalHeight = maximumHeight;

double startX = minimumWidthMargin;

double startY = minimumHeightMargin;

if (finalWidth > maximumWidth) {

finalHeight = maximumWidth \* TRIANGLE\_HEIGHT / TRIANGLE\_WIDTH;

finalWidth = maximumWidth;

startX = minimumWidthMargin + (maximumWidth - finalWidth) / 2.0;

} else {

startY = minimumHeightMargin + (maximumHeight - finalHeight) / 2.0;

}

int[] xCoordinates = new int[3];

int[] yCoordinates = new int[3];

xCoordinates[0] = (int) startX;

xCoordinates[1] = xCoordinates[0] + (int) finalWidth;

xCoordinates[2] = xCoordinates[1];

yCoordinates[0] = (int) startY;

yCoordinates[1] = yCoordinates[0];

yCoordinates[2] = yCoordinates[0] + (int) finalHeight;

userTriangle = new Polygon(xCoordinates, yCoordinates, 3);

}

Il poligono di taglio invece viene calcolato tramite i punti che si trovano sul triangolo di modello e il triangolo che sta utilizzando l'utente. Per fare questa magia il programma sfrutta semplicemente la proporzione tra i die triangoli.

/\*\*

\* Calcola e ritorna il poligono personalizzato.

\*

\* @param windowWidth Larghezza della finestra.

\* @param windowHeight Altezza della finestra.

\* @return Il poligono già pronto da disegnare.

\*/

public Polygon getCustomPolygon(int windowWidth, int windowHeight) {

calculateTriangleFromWindowSize(windowWidth, windowHeight);

int[] xCoordinates = new int[customPolygonDots.size()];

int[] yCoordinates = new int[customPolygonDots.size()];

for (int i = 0; i < customPolygonDots.size(); i++) {

xCoordinates[i] = userTriangle.getBounds().x + (int) (customPolygonDots.get(i).getX() \* userTriangle.getBounds().width / TRIANGLE\_WIDTH);

yCoordinates[i] = userTriangle.getBounds().y + (int) (customPolygonDots.get(i).getY() \* userTriangle.getBounds().height / TRIANGLE\_HEIGHT);

}

return new Polygon(xCoordinates, yCoordinates, customPolygonDots.size());

}

## Aggiunta e rimozione dei punti

Anche l'aggiunta dei punti è completamente gestita dalla classe TriangleModel che richiede come parametro il punto che si vuole aggiungere e la dimensione della finestra. Sapendo queste informazioni il TriangleModel richiamerà il metodo calculateTriangleFromWindowSize, in modo da sapere su che triangolo sta operando, e proporizonerà in automatico i punti.

/\*\*

\* Date le dimensioni della finestra ricalcola un punto in modo da adattarlo

\* al triangolo di modello.

\*

\* @param dot Punto da adattare.

\* @param width Larghezza della finestra.

\* @param height Altezza della finestra.

\* @return Il punto adattato.

\*/

private Point getModelPoint(Point dot, int width, int height) {

calculateTriangleFromWindowSize(width, height);

double x = dot.getX() - userTriangle.xpoints[0];

double y = dot.getY() - userTriangle.ypoints[0];

double finalX = x \* TRIANGLE\_WIDTH / userTriangle.getBounds().width;

double finalY = y \* TRIANGLE\_HEIGHT / userTriangle.getBounds().height;

return new Point((int) finalX, (int) finalY);

}

/\*\*

\* Aggiunge un punto al modello.

\*

\* @param dot Punto da aggiungere.

\* @param width Larghezza della finestra.

\* @param height Altezza della finestra.

\*/

public void addDotToCustomPolygon(Point dot, int width, int height) {

customPolygonDots.add(getModelPoint(dot, width, height));

}

La rimozione dei punti invece richiede un lavoro aggiuntivo alla classe TrianglePanel (contenente il TriangleModel), perché deve controllare se il mouse è all'interno dell'area del punto e passare come posizione non quella effettiva ma quella del punto sopra al quale ci si trova (la stessa tecnica viene utilizzata per muovere i punti).

/\*\*

\* Dice se un punto esiste nella lista di punti già presenti.

\*

\* @param position Punto da controllare.

\* @return Punto esistente, altrimenti 0,0.

\*/

private Point getPointInteressed(Point position) {

Point res = new Point(0, 0);

for (Point dot : dots) {

if (dot.distance(position) <= POINT\_RADIUS \* 2) {

return dot;

}

}

return res;

}

public void mouseClicked(MouseEvent e) {

if (addPoint) {

Point p = e.getPoint();

triangleModel.addDotToCustomPolygon(p, this.getWidth(), this.getHeight());

dotInteressed = dots.size();

repaint();

addedAPoint();

} else {

Point p = getPointInteressed(e.getPoint());

if (p.x != 0 && p.y != 0) {

triangleModel.removeDotFromPolygon(p, this.getWidth(), this.getHeight());

repaint();

addedAPoint();

}

}

}

## Salvataggio e importazione dei punti

I punti vengono salvati su un seplicissimo csv e la loro formattazione è in base al triangolo di modello, che è una costante all'interno della classe TriangleModel. Facendo questo approccio non c'è bisogno di salvarsi la dimensione della finestra al momento del salvataggio perché si saprà già al momento dell'apertura che sono stati salvati in base al triangle model.

## Creazione del fiocco di neve

Il fiocco di neve viene generato partendo dal TriangleModel. Da esso viene sottratto l'area del poligono di taglio e poi viene specchiato (in caso che il triangolod debba essere specchiato) e girato per creare il fiocco.

private Shape rotateArea(int angle, Shape shape) {

AffineTransform af = new AffineTransform();

af.rotate(Math.toRadians(angle), triangle.xpoints[2], triangle.ypoints[2]);

return af.createTransformedShape(shape);

}

/\*\*

\* Permette di avere l'area specchiata.

\*

\* @return Il triangolo specchiato.

\*/

private Shape getFlippedArea(Area area) {

AffineTransform first = new AffineTransform();

first.scale(-1, 1);

AffineTransform toCenter = new AffineTransform();

toCenter.translate(-(triangle.xpoints[1]) \* 2, 0);

AffineTransform tot = new AffineTransform();

tot.concatenate(first);

tot.concatenate(toCenter);

return tot.createTransformedShape(area);

}

Una volta che si hanno questi due metodi non basta altro che usufruirne all'interno del metodo paint nella seguente maniera:

for (int i = 0; i < 6; i++) {

g2.fill(rotateArea((i - 1) \* 60, getFlippedArea(finalArea)));

g2.fill(rotateArea(i \* 60, finalArea));

}

## Cambiamento di design

\* \*\*Pagina iniziale\*\*: La pagina iniziale non esiste più, questa scelta è stata intrapresa a causa del fatoo che sarebbe stata inutile. Per rendere il programma meno complicato per l'utente questa schermata è stata assorbita dalla pagina principale (dove si modifica il triangolo).

\* \*\*Pagina principale\*\*: Nella pagina principale sono state aggiunte diverse funzioni, che sono necessarie per il corretto funzionamento del programma. I 2 menù non sono più divisi solo da un lato ma sono completamente separati (ora i bottoni che interagiscono con i punti sono a sinistra e gli altri a destra). Tra le cose aggiunte troviamo

1. \*undo\* per eliminare l'ultimo punto

2. \*Tutorial\* mostra un mini tutorial all'utente sulle funzioni del programma

3. \*live-preview\* è stata eliminata per renderla sempre attiva quando si genera il fiocco, infatti non si trova più in basso a destra ma in un'altra finestra (nel primo caso risultava molto piccolo e difficile da capire).

4. \*nome del triangolo corrente\* sempre in basso a sinistra

5. \*posizione del mouse\* in pixel e sempre leggibile in basso

\* \*\*Pagina di salvataggio\*\*: La finestra di salvataggio è diventata la classica di Windows (per semplicità) ma affiancata dalla preview di quello che stiamo per salvare. Quando salviamo i punti la preview ci farà vedere il triangolo tagliato, mentre quando salviamo il fiocco la preview mostrerà il fiocco generato.

\* \*\*Sito Web\*\*: come potevo dedurre dall'inizio il sito web è cambiato drasticamente, adesso è molto più esteso, con una guida, i requisiti e un'apposita finestra per il download. Oltre a tutto ciò è stato aggiunto anche un form per fare in modo che gli utenti possano chiedere aiuto all'autore o riferirgli dei bug

# Test

## Protocollo di test

|Test Case | TC-001 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Provare programma |

|\*\*Riferimento\*\*|REQ-001 |

|\*\*Descrizione\*\*|Scaricare il programma dal sito ufficiale e provarlo|

|\*\*Prerequisiti\*\*|Avere la base del sito web e il programma comlpetamente funzionante|

|\*\*Procedura\*\* |Recarsi sul sito ufficiale, scaricarlo e provare le funzioni principali|

|\*\*Risultati attesi\*\* |Il programma dovrebbe funzionare normalmente come se avviato da NetBeans|

|Test Case | TC-002 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Ridimensionare finestra|

|\*\*Riferimento\*\*|REQ-004 |

|\*\*Descrizione\*\*|Controllare se si può diminuire la dimensione del programma sotto a 1024x768 pixel|

|\*\*Prerequisiti\*\*|Impostare la dimensione minima|

|\*\*Procedura\*\* |Avviare programma, ridimensionare in piccolo. Se funziona: andare nelle impostazioni della macchina con il programma aperto e impostare la risoluzione dello schermo inferiore a 1024x768|

|\*\*Risultati attesi\*\* |Il programma dovrebbe rimanere della dimensione minima e nel secondo caso sforare dallo schermo|

|Test Case | TC-003 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Aggiungere punti|

|\*\*Riferimento\*\*|REQ-006 |

|\*\*Descrizione\*\*|Provare ad aggiungere punti al triangolo|

|\*\*Prerequisiti\*\*|Implementare la possibilità di aggiungere i punti|

|\*\*Procedura\*\* |Avviare programma, cliccare sul panel e al di fuori per aggiungere dei punti|

|\*\*Risultati attesi\*\* |I punti dovrebbero aggiungersi unicamente all'interno del panel senza sforare|

|Test Case | TC-004 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Reset punti|

|\*\*Riferimento\*\*|REQ-008 |

|\*\*Descrizione\*\*|Testare il bottone reset|

|\*\*Prerequisiti\*\*|Implementare la possibilità di aggiungere i punti e il reset dei punti|

|\*\*Procedura\*\* |Avviare programma, aggiungere punti, cliccare reset, aggiungere altri punti|

|\*\*Risultati attesi\*\* |I primi punti si dovranno eliminare tranquillamente e quando se ne aggiungeranno poi di nuovi i vecchi non dovranno ricomparire|

|Test Case | TC-005 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Switch add e cut|

|\*\*Riferimento\*\*|REQ-009 |

|\*\*Descrizione\*\*|Testare lo switch tra add e cut|

|\*\*Prerequisiti\*\*|Implementare la possibilità di aggiungere i punti e rimuoverli|

|\*\*Procedura\*\* |Avviare programma, aggiungere punti, cliccare cut, eliminare punti esistenti, cliccare su add, aggiungere punti|

|\*\*Risultati attesi\*\* |I punti si dovrebbero aggiungere e rimuovere tranquillamente|

|Test Case | TC-006 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Drag punti|

|\*\*Riferimento\*\*|REQ-010 |

|\*\*Descrizione\*\*|Testare il drag dei punti|

|\*\*Prerequisiti\*\*|Implementare la possibilità di aggiungere i punti e muoverli|

|\*\*Procedura\*\* |Avviare programma, aggiungere punti, tenere premuto su uno e muoverlo|

|\*\*Risultati attesi\*\* |I punti si dovrebbero spostare tranquillamente senza crearne degli altri|

|Test Case | TC-007 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Anteprima live|

|\*\*Riferimento\*\*|REQ-011 |

|\*\*Descrizione\*\*|Testare la generazione live|

|\*\*Prerequisiti\*\*|Implementare la generazione del fiocco|

|\*\*Procedura\*\* |Avviare programma, aggiungere punti, generare fiocco, aggiungere nuovi punti, muoverne alcuni, rimuoverne altri e alla fine fare un reset|

|\*\*Risultati attesi\*\* |I punti dovrebbero rispondere correttamente ai comandi senza sollevare nessun problema|

|Test Case | TC-008 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Salvataggio fiocco|

|\*\*Riferimento\*\*|REQ-013 |

|\*\*Descrizione\*\*|Salvare il fiocco|

|\*\*Prerequisiti\*\*|Implementare il salvataggio del fiocco|

|\*\*Procedura\*\* |Avviare programma, aggiungere punti, salvare il fiocco|

|\*\*Risultati attesi\*\* |Il fiocco si dovrebbe salvare|

|Test Case | TC-009 |

|---------------|--------------------------------------|

|\*\*Nome\*\* |Snapshot|

|\*\*Riferimento\*\*|REQ-016 |

|\*\*Descrizione\*\*|Giocare con gli snapshot|

|\*\*Prerequisiti\*\*|Implementare il salvataggio dei punti|

|\*\*Procedura\*\* |Avviare programma, aggiungere punti, salvare lo snapshot, chiudere il programma, aprire il programma, aprire snapshot|

|\*\*Risultati attesi\*\* |Il programma dovrebbe riaprire il primo triangolo con i relativi tagli|

## Risultati test

|Test|Risultato|

|---|--|

|001|Successo|

|002|Successo|

|003|Successo|

|004|Successo|

|005|Successo|

|006|Successo|

|007|Successo|

|008|Successo|

|009|Successo|

## Mancanze/limitazioni conosciute

\* \*\*Salvataggio in PNG\*\*: Il salvataggio in formato PNG non è stato possibile a causa del tempo esaurito.

\* \*\*Possibilità di usare più poligoni di taglio\*\*: Requisito non rispettato a causa del fatto che la struttura del programma non permetteva questa opzione.

## Consuntivo

Il tempo di lavoro effettivo era giusto, anzi quasi abbondante. Il problema è stata la progettazione fatta in maniera poco curata e rispettata estremamente poco.

# Conclusioni

## Sviluppi futuri

Magari l'aggiunta dei requisiti non rispettati.

## Considerazioni personali

Grazie a questo progetto ho capito quanto una buona progettazione possa rendere un progetto molto più facile da realizzare. In questo progetto purtroppo non sono riuscito a gestirlo benissimo ma tutto quello che ho imparato verrà sicuramente usato per progetti futuri.