Documentazione Mouse Mover

# Indice

1 Indice 2

2 Analisi 2

2.1 Analisi del dominio 2

2.2 Analisi e specifica dei requisiti 3

2.3 Use case 5

2.4 Pianificazione 6

2.5 Analisi dei mezzi 7

2.5.1 Software 7

2.5.2 Hardware 7

3 Progettazione 8

3.1 Design dell’architettura del sistema 8

3.2 Design dei dati e database 8

3.3 Design delle interfacce 8

3.4 Design procedurale 9

4 Implementazione 10

4.1 Preparazione 10

4.2 Background trasparente 10

4.3 Primo piano 11

4.4 Controllo movimento mouse 11

4.5 Animazione step by step 12

4.6 Menu 13

4.7 Input offset temporale 14

4.8 Icona nella system tray 15

4.9 Cambio cursore 15

5 Test 16

5.1 Protocollo di test 16

5.2 Risultati test 18

5.3 Mancanze/limitazioni conosciute 18

6 Consuntivo 19

7 Conclusioni 20

7.1 Sviluppi futuri 20

7.2 Considerazioni personali 20

8 Bibliografia 21

8.1 Sitografia 21

9 Glossario 22

10 Indice delle figure 22

11 Allegati 22

# Analisi

## Analisi del dominio

Questo programma viene usato singolarmente su un pc con windows, ed è principalmente utile ad utenti che devono spesso staccarsi dallo schermo ma non vogliono che esso si spenga.  
Un suo altro grande potenziale di utilizzo è durante lunghi download, dove l’utente non si dovrà più preoccupare di mantenere lo schermo attivo.

Per utilizzare l’applicazione non è necessario nessun tipo di prerequisito, l’unica cosa da sapere è che il menu è accessibile solo dal system tray.

## Analisi e specifica dei requisiti

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-1 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Eseguire le animazioni ogni x minuti | | | | |
| **Note** | x verrà definito nel menu come offset temporale | | | | |
| **Sotto requisiti** | | | | | |
| **001** | Creare un menu | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-2 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Scegliere cursore mouse | | | | |
| **Note** | Sarà presente una cartella in cui poterli caricare | | | | |
| **Sotto requisiti** | | | | | |
| **001** | Possibilità di cambiare cursore durante il programma | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-3 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Sospendere il programma | | | | |
| **Note** | Una volta sospeso il programma potranno essere applicati i sotto requisiti del Req-2 | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requisito** | Req-4 | **Priorità** | 1 | **Versione** | 1.0 |
| **Nome** | Riprendere/Chiudere programma | | | | |
| **Note** | Necessario il Req-3 | | | | |

**Spiegazione elementi tabella dei requisiti:**

**Req**: identificativo univoco del requisito

**Nome**: nome del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

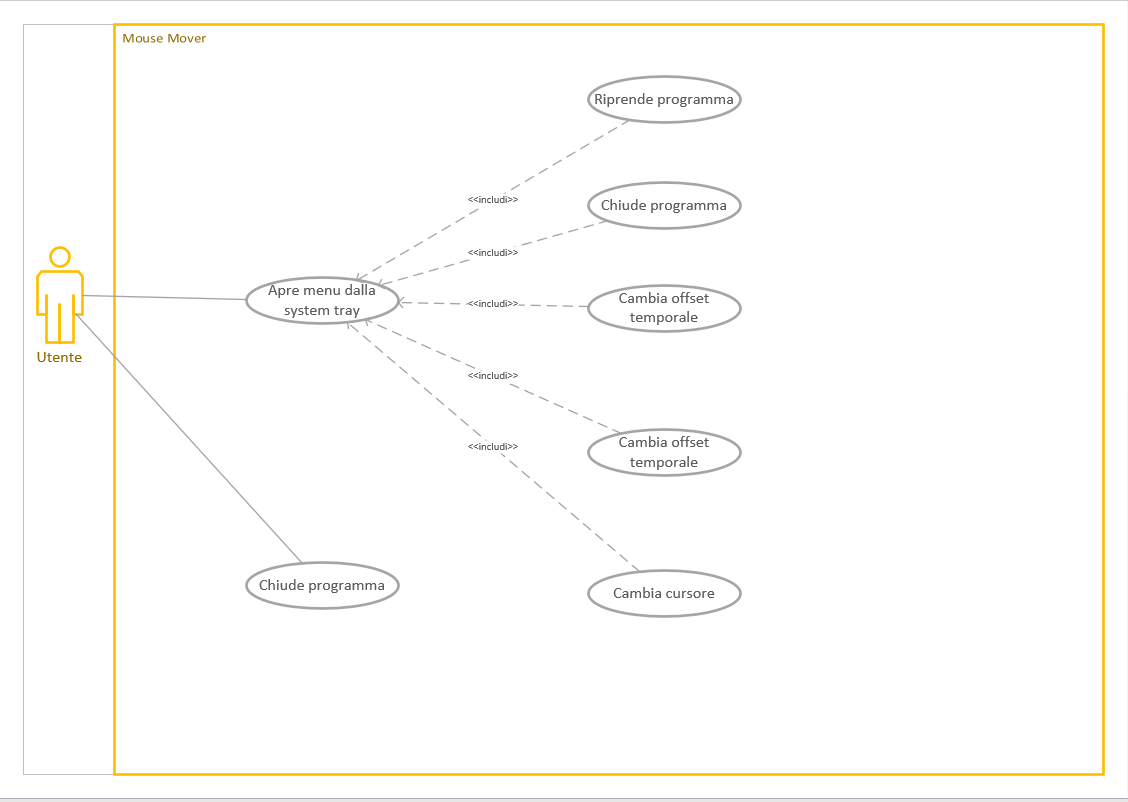
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

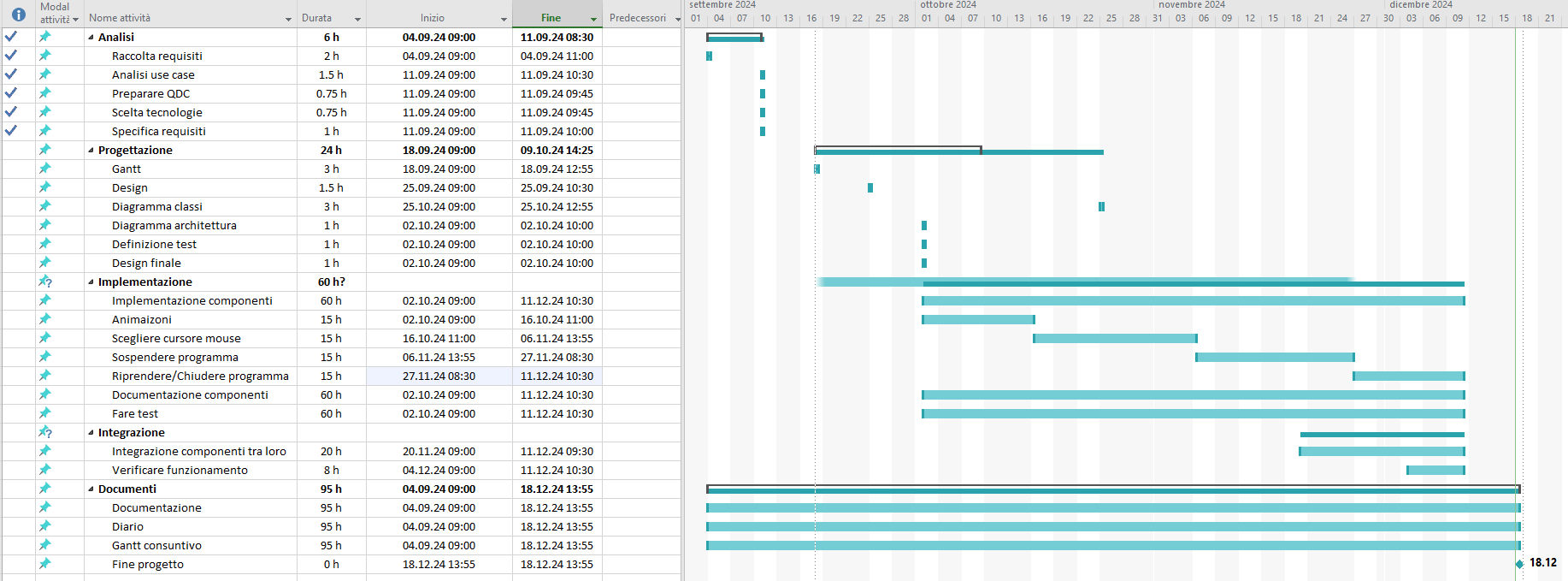
## Use case



Figura

Una volta avviato il programma l’utente può solo accedere o chiudere il programma dal system tray. Se viene aperto il menu può riprendere il programma, chiuderlo, cambiare l’offset temporale delle animazioni o cambiare il cursore.

## Pianificazione



Figura

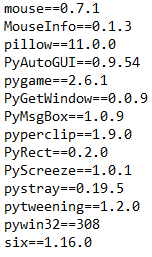
## Analisi dei mezzi

Per lo sviluppo del progetto avevo a disposizione il computer della scuola, internet e l’aiuto da parte del docente supervisore Geo Petrini.

### Software

* Python 3.10

Requirements:



### Hardware

Il programma è utilizzabile su un qualsiasi PC con Windows.

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

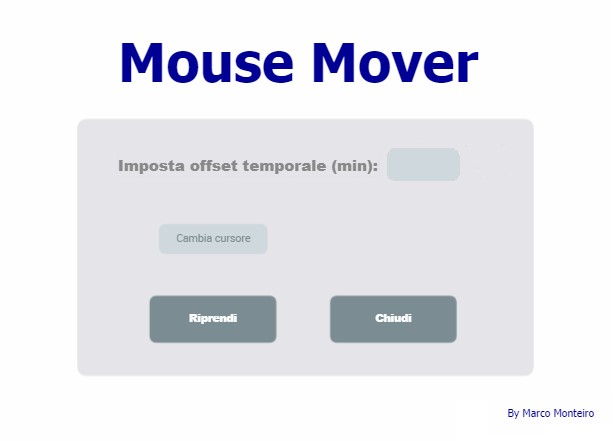
## Design dell’architettura del sistema

Essendo un programma piccolo tutto il sistema viene gestito in una cartella locale, compresi i cursori disponibili.

## Design dei dati e database

Essendo questo progetto piccolo e quindi avendo pochi cursori e animazioni disponibili tutte le risorse vengono gestite in locale.

## Design delle interfacce



Figura

Sarà possibile cliccare ogni bottone eseguendo così la sua funzione, per l’offset temporale sarà possibile inserire un input.

## Design procedurale

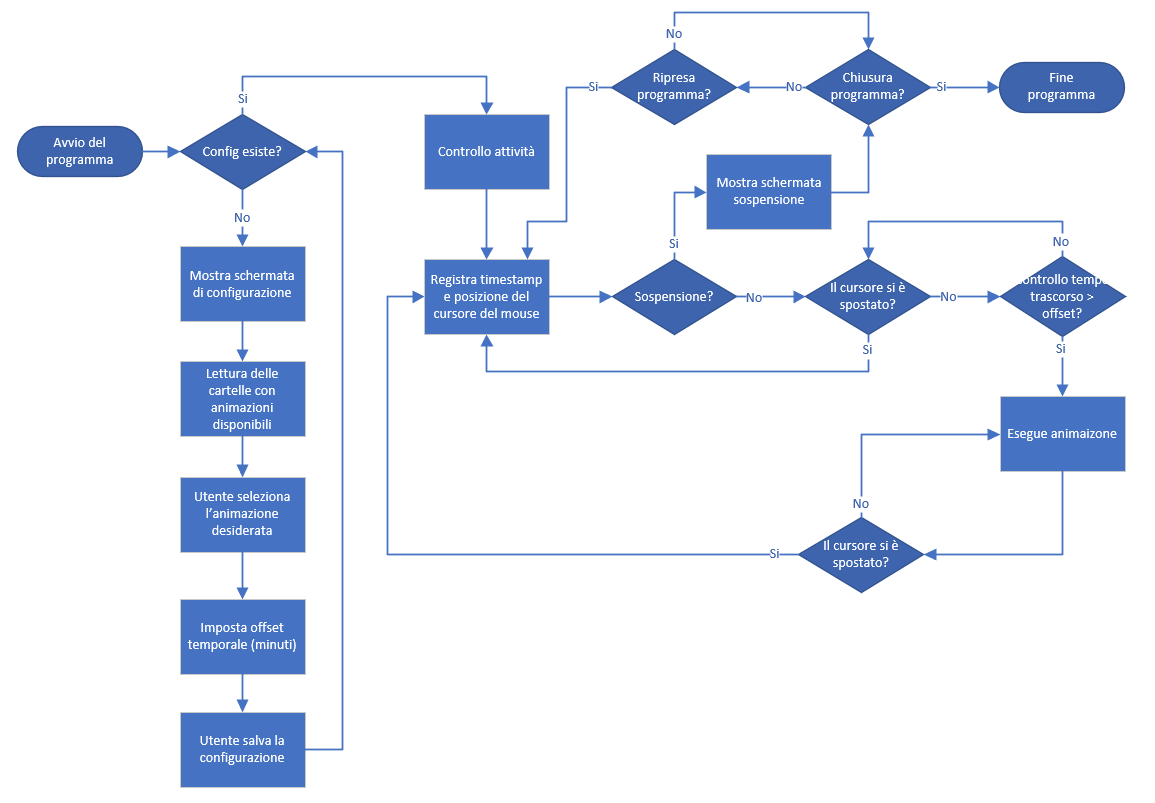


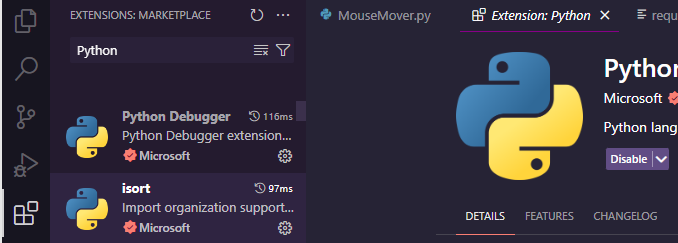
Figura 4

# Implementazione

## Preparazione

Come prima cosa bisognerà creare una cartella per il progetto, una volta fatto sarà necessario  
aprire “Visual Studio Code” o un qualsiasi IDE a piacimento, e usare Ctrl+K+O per aprire la cartella creata in precedenza.

Una volta fatto bisognerà scaricare Python dalla sezione apposita.



Figura

Ora bisognerà aprire un nuovo terminale nella navbar in alto,

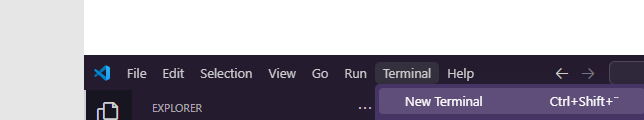


Figura 6

ed eseguire i seguenti comandi (controllare di essere nella cartella in cui c’è il programma):

* “pip install pygame”
* “pip install pyautogui”
* “pip install pystray”

## Background trasparente

Pygame non supporta direttamente la trasparenza, quindi per implementarla è stato necessario creare una finestra che supportasse la trasparenza, colorarla interamente con un qualsiasi colore, e in seguito utilizzare le librerie “Win32gui” e “win32con” per definire quel colore come trasparente.

key\_color\_hex = 0x010101 *# Key color to make transparent*

*self*.screen = pygame.display.set\_mode((width, height), pygame.NOFRAME | pygame.SRCALPHA) *# Create a borderless window with alpha support*

win32gui.SetWindowLong(*self*.hwnd, win32con.GWL\_EXSTYLE, win32gui.GetWindowLong(*self*.hwnd, win32con.GWL\_EXSTYLE) | win32con.WS\_EX\_LAYERED) *# Set the window as "layered"*

        win32gui.SetLayeredWindowAttributes(*self*.hwnd, *self*.key\_color\_hex, 0, win32con.LWA\_COLORKEY) *# Set the background transparency*

Come è possibile notare è anche necessario impostare la finestra come “layered”, cioè a più livelli di trasparenza o effetti.

Per andare a interagire con la finestra viene usato il suo hwnd(handle), che è un identificatore univoco usato dai sistemi operativi per fare riferimento alla finestra di un’applicazione.

## Primo piano

Il cursore e il timer del programma devono rimanere sempre visibili, indipendentemente dalla finestra utilizzata. Per farlo è stato necessario impostare la finestra dell’applicazione in primo piano.

win32gui.SetWindowPos(*self*.hwnd, win32con.HWND\_TOPMOST, 0, 0, 0, 0, win32con.SWP\_NOMOVE | win32con.SWP\_NOSIZE)

Questa funzione va a modificare la posizione dell’applicazione rispetto alla gerarchia delle finestre, i parametri sono: finestra, ordine, x e y, uFlags.

Per impostarla in primo piano nel parametro odine viene assegnato il valore HWND\_TOPMOST, mentre le flag utilizzate indicano che la finestra non viene spostata o ridimensionata.

## Controllo movimento mouse

def refreshMousePosition(*self*):

*self*.mouse\_x, *self*.mouse\_y = pyautogui.position()

    def mouseController(*self*):

        if not *self*.displayingMenu:

            if *self*.initialPosition != pyautogui.position():

*self*.initialPosition = pyautogui.position()

*self*.initialTime = time.time()

            elif time.time() - *self*.initialTime > *self*.temporalOffset:

*self*.runningAnimation = True

*self*.moveMouseSquare()

*self*.mouseMoveCircle()

*self*.initialTime = time.time()

Questa funzione verifica costantemente se la posizione del mouse è diversa da quella precedete.

In caso sia diversa imposta nuovo tempo e posizione iniziali.  
In caso la posizione sia uguale a quella precedente controlla se il tempo trascorso dal tempo iniziale è maggiore dell’offset temporale impostato, e in caso lo sia, esegue le animazioni assegnate. La posizione del mouse viene catturata attraverso “pyautogui.position()”, che è un metodo della libreria “pyautogui”.

## Animazione step by step

Inizialmente le animazioni erano un blocco intero di codice, ma ciò aveva degli intoppi, ovvero durante l’esecuzione dell’animazione la posizione del cursore e la posizione e contenuto del timer non vengono aggiornati.  
Per risolvere il problema ho spezzettato le animazioni, e dopo ogni pezzo ho eseguito un refresh generale.

def moveMouseSquare(*self*):

        size = 200

*# Moves the mouse simulating a square*

        for i in range(0, size, 1):

            if not *self*.runningAnimation:

                break

            mouse.move(1, 0, *absolute*=False, *duration*=0.001)

*self*.loadAll()

*self*.isInterrupted(*self*.mouse\_x, *self*.mouse\_y)

        for i in range(0, size, 1):

            if not *self*.runningAnimation:

                break

            mouse.move(0, 1, *absolute*=False, *duration*=0.001)

*self*.loadAll()

*self*.isInterrupted(*self*.mouse\_x, *self*.mouse\_y)

        for i in range(0, size, 1):

            if not *self*.runningAnimation:

                break

            mouse.move(-1, 0, *absolute*=False, *duration*=0.001)

*self*.loadAll()

*self*.isInterrupted(*self*.mouse\_x, *self*.mouse\_y)

        for i in range(0, size, 1):

            if not *self*.runningAnimation:

                break

            mouse.move(0, -1, *absolute*=False, *duration*=0.001)

*self*.loadAll()

*self*.isInterrupted(*self*.mouse\_x, *self*.mouse\_y)

Durante l’esecuzione dell’animazione viene anche verificato se il mouse viene mosso dall’utente, e in quel caso termina automaticamente l’animazione.

## Menu

def displayMenu(*self*):

        if *self*.displayingMenu:

            info = pygame.display.Info()

            menu = pygame.image.load('GUI/GUI principale.png')

*self*.screen.blit(menu, (info.current\_w / 2 - 300, info.current\_h / 2 - 200))

            temporalOffsetText = *self*.font.render(str(*self*.temporalOffset), True, (136,136,136))

            if *self*.temporalOffset < 10:

*self*.screen.blit(temporalOffsetText, (info.current\_w / 2 + 115, info.current\_h / 2 - 45))

            else:

*self*.screen.blit(temporalOffsetText, (info.current\_w / 2 + 109, info.current\_h / 2 - 45))

*self*.setToForeground()

*self*.menuController(info)

Con questo codice viene semplicemente caricata l’immagine del menu e poi richiamata la funzione “menuController” che va a gestire tutti i “bottoni” all’interno dell’immagine attraverso dei range di coordinate.

## Input offset temporale

def handleInput(*self*, *info*, *typing*):

        input\_text = ""

        input\_rect = pygame.Rect(*info*.current\_w / 2 + 105, *info*.current\_h / 2 - 52, 50, 30)

        while *typing*:

            for event in pygame.event.get():

                if event.type == pygame.KEYDOWN:

                    if event.key == pygame.K\_RETURN:

                        if input\_text != "":

*self*.temporalOffset = int(input\_text) *# \*60 per i minuti*

*typing* = False

                    elif event.key == pygame.K\_BACKSPACE:

                        input\_text = input\_text[:-1]  *# Remove the last character*

                    elif len(input\_text) < 2 and '0' <= event.unicode <= '9':  *# Limita i numeri a 1-9 e due cifre*

                        input\_text += event.unicode  *# Add the typed character to the input text*

            menu = pygame.image.load('GUI/GUI principaleInput.png')

*self*.screen.blit(menu, (*info*.current\_w / 2 - 300, *info*.current\_h / 2 - 200))

            color = (123,141,147)

            pygame.draw.rect(*self*.screen, color, input\_rect)

            text\_surface = *self*.font.render(input\_text, True, (0,0,0))

            if len(input\_text) == 1:

*self*.screen.blit(text\_surface, (input\_rect.x + 11, input\_rect.y + 6))

            else:

*self*.screen.blit(text\_surface, (input\_rect.x + 5, input\_rect.y + 6))

            pygame.display.flip()

Per gestire l’input viene creato un Rect, all’interno del quale viene costantemente inserito il contenuto di “input\_text”. Mentre si sta scrivendo il programma controlla se vengono premuti dei tasti:

* Se viene premuto “Enter” e input\_text non è vuoto viene salvato il nuovo offset temporale.
* Se viene premuto “Backspace” viene cancellato l’ultimo carattere inserito.
* Se vengono inseriti numeri da 0-9 e “input\_text” ha meno di due cifre viene inserita la nuova cifra in “input\_text”.

Infine viene disegnato il Rect con all’interno “input\_text”, che viene posizionato in base al numero di cifre per allinearlo al centro.

## Icona nel system tray

Per scelta personale ho voluto togliere l’icona del programma dalla barra delle applicazioni e inserirla nel system tray.

win32gui.SetWindowLong(*self*.hwnd, win32con.GWL\_EXSTYLE, win32gui.GetWindowLong(*self*.hwnd, win32con.GWL\_EXSTYLE) | win32con.WS\_EX\_TOOLWINDOW)

Questo codice prende lo stile della finestra e ne va a impostare l’applicazione come “toolwindow”, in modo che l’icona venga automaticamente rimossa dalla barra delle applicazioni.

Per gestire l’icona nel system tray viene startata una thread.

def start\_tray(*self*):

        icon\_image = *self*.create\_image()  *# Uses the icon created by the function*

*self*.icon = Icon("test\_icon", icon\_image, *menu*=Menu(MenuItem('Apri menu', *self*.loadMenu), MenuItem('Chiudi', *self*.on\_quit)))

*self*.icon.run()

Qua vengono assegnati all’icona un’immagine e un menu dropdown che si attiva col click destro sull’icona. Ad ogni elemento disponibile nel dropdown viene assegnato un rispettivo metodo che viene invocato al click.

## Cambio cursore

def findCursor(*self*):

        root = tk.Tk()

        root.withdraw()  *# Nasconde la finestra principale di tkinter*

*# Definisci un percorso relativo, ad esempio nella cartella 'cursors'*

        percorso\_relativo = os.path.join(os.getcwd(), 'Cursors')

*self*.cursorPath = filedialog.askopenfilename(*title*="Seleziona un'immagine del cursore", *initialdir*=percorso\_relativo, *filetypes*=[("Immagini PNG", "\*.png")])

*self*.showingCursorSelector = False

        return *self*.cursorPath

Per poter cambiare cursore a partire da un’immagine (caricabile anche sul momento), bisognerà importare la libreria “tkinter”.

In questo metodo viene creato un oggetto di tipo “Tk“ e poi viene nascosta la finestra principale di tkinter. Crearlo permette di aprire una finestra per la ricerca di file.

Viene poi creato un percorso relativo unendo “os.getcwd()” a ‘Cursors’.  
“os.getcwd()” restituisce il percorso assoluto della cartella corrente, mentre “os.path.join()” si occupa di unire i due percorsi in base alle convenzioni del sistema operativo utilizzato, esempio:  
Windows: Il risultato sarà🡪”percorsoAssoluto\Cursors”.

Linux/macOS: Il risultato sarà🡪”percorsoAssoluto/Cursors”.

Una volta creato il percorso, esso verrà utilizzato come cartella in cui cercare file attraverso il metodo “filedialog.askopenfilename”, in cui viene definito che sono accettati solo file png.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-01 | **Nome** | Eseguire le animazioni ogni x minuti |
| **Descrizione** | Le animazioni dovranno essere eseguite ogni x minuti | | |
| **Prerequisiti** | Avere un menu in cui è possibile modificare l’offset temporale | | |
| **Procedura** | 1. Avviare il programma 2. Aprire il system tray 3. Fare click destro sull’icona del programma (shrek) 4. Cliccare “Apri menu” 5. Inserire nel riquadro apposito il nuovo offset temporale 6. Riprendere il programma | | |
| **Risultati attesi** | Offset temporale dell’animazione cambiato | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-02 | **Nome** | Scegliere cursore mouse |
| **Descrizione** | Cambiamento del cursore | | |
| **Prerequisiti** | Avere un menu in cui è possibile cambiare il cursore | | |
| **Procedura** | 1. Avviare il programma 2. Aprire il system tray 3. Fare click destro sull’icona del programma (shrek) 4. Cliccare “Apri menu” 5. Cliccare il “bottone” apposito 6. Selezionare il nuovo cursore 7. Confermare selezione | | |
| **Risultati attesi** | Cursore cambiato | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-03 | **Nome** | Sospendere il programma |
| **Descrizione** | Il programma viene sospeso e viene aperto il menu | | |
| **Prerequisiti** | Aver cliccato “Apri menu” | | |
| **Procedura** | 1. Avviare il programma 2. Aprire il system tray 3. Fare click destro sull’icona del programma (shrek) 4. Cliccare “Apri menu” | | |
| **Risultati attesi** | Programma sospeso e menu aperto | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-04 | **Nome** | Riprendere/Chiudere programma |
| **Descrizione** | Il programma viene ripreso/chiuso | | |
| **Prerequisiti** | Essere nel menu oppure aver aperto il dropdown dell’icona nel system tray | | |
| **Procedura** | 1. Avviare il programma 2. Aprire il system tray 3. Fare click destro sull’icona del programma (shrek) 4. Cliccare “Chiudi”   Oppure:   1. Cliccare “Apri menu” 2. Cliccare il “bottone” con scritto “Chiudi” oppure “Riprendi” | | |
| **Risultati attesi** | Programma terminato/ripreso | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 | Figura 7 | Passato |
| TC-002 | Figura 8 | Passato |
| TC-003 | Figura 9 | Passato |
| TC-004 | Figura 10 | Passato |

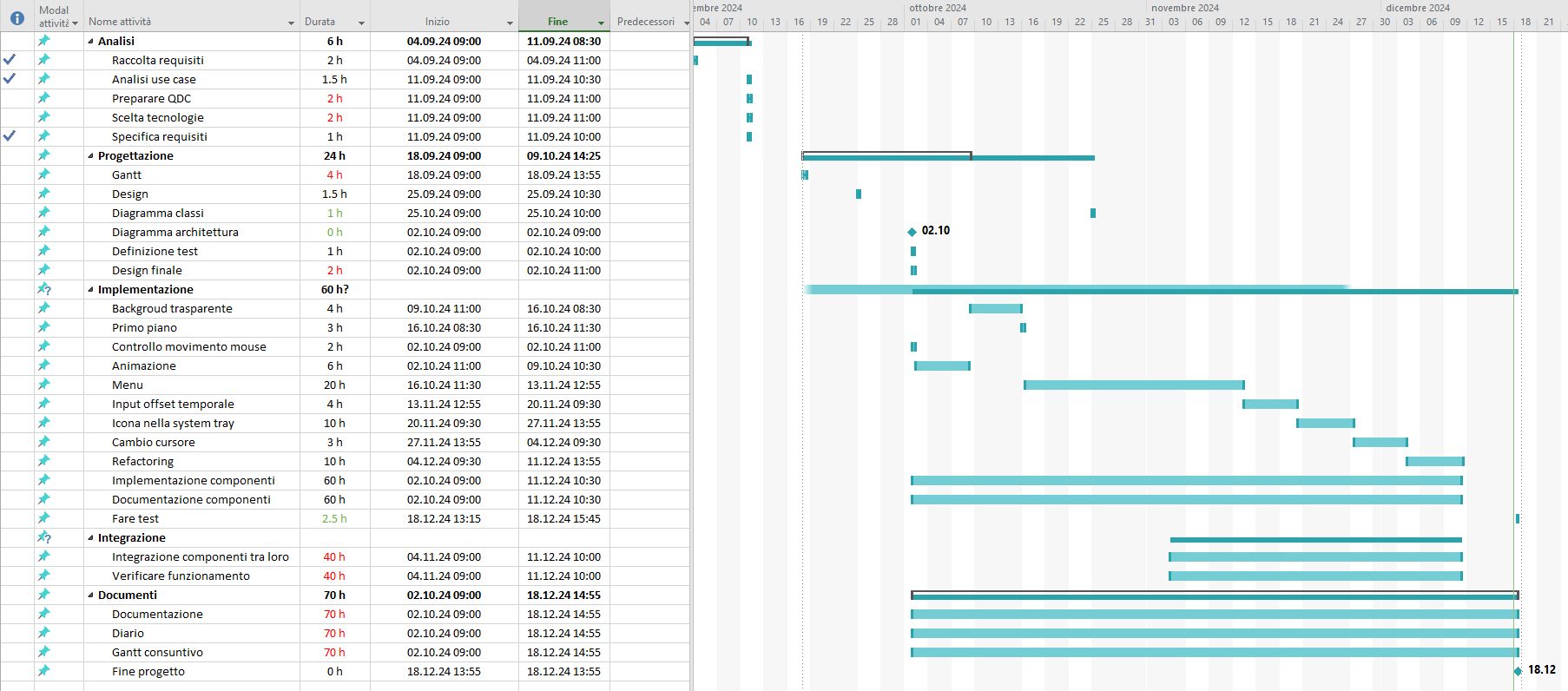
## Mancanze/limitazioni conosciute

Non ho svolto nessuno dei requisiti opzionali perché consultando il cliente ho avuto nuove idee che mi sono piaciute di più e che ho trovato più utili per il funzionamento del programma.

Requisiti opzionali tralasciati:

* Funzionare su più schermi
* Possibilità di cambiare animazione
* Poter selezionare l’animazione corrente
* Riprodurre suoni durante le animazioni

# Consuntivo



Figura

# Conclusioni

Questo progetto mi è piaciuto molto perché risolve un problema comune che io ritrovo spesso quando utilizzo un computer.  
D’ora in poi lo utilizzerò sicuramente quando dovrò svolgere dei lunghi download, così potrò stare tranquillo senza dover tenere sott’occhio lo schermo.

Non penso che sia di estrema importanza, ma penso che con ulteriori sviluppi e miglioramenti abbia un grande potenziale.

## Sviluppi futuri

* Funzionare su più schermi
* Possibilità di cambiare animazione
* Riprodurre suoni durante le animazioni

## Considerazioni personali

Questo progetto mi è stato molto utile per accrescere le mie conoscenze di base in python e pygame, e mi è piaciuto molto il modo in cui sono dovuto andare a ragionare per risolvere le varie problematiche importanti. Penso che in un futuro progetto, anche se non inerente a questo, avrei una mentalità e un modo di ragionare completamente diversi, e questo mi accresce molto a livello professionale.

# Bibliografia

## Sitografia

1. <https://www.youtube.com/>
2. <https://chatgpt.com/>
3. <https://www.pygame.org/docs/>
4. <https://www.w3schools.com/>

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| Offset temporale | Corrisponde al tempo che il mouse deve stare fermo per eseguire l’animaizone |
| System tray | Sezione della barra delle applicazioni situata in basso a destra |

# Indice delle figure

[Figura 1 5](#_Toc185427962)

[Figura 2 6](#_Toc185427963)

[Figura 3 8](#_Toc185427964)

[Figura 4 9](file:///D:\Scuola\GitHub\ProgettoM306\3_Documentazione\Documentazione%20Mouse%20Mover.docx#_Toc185427965)

[Figura 5 10](#_Toc185427966)

[Figura 6 10](file:///D:\Scuola\GitHub\ProgettoM306\3_Documentazione\Documentazione%20Mouse%20Mover.docx#_Toc185427967)

[Figura 7 18](file:///D:\Scuola\GitHub\ProgettoM306\3_Documentazione\Documentazione%20Mouse%20Mover.docx#_Toc185427968)

[Figura 8 18](file:///D:\Scuola\GitHub\ProgettoM306\3_Documentazione\Documentazione%20Mouse%20Mover.docx#_Toc185427969)

[Figura 9 18](file:///D:\Scuola\GitHub\ProgettoM306\3_Documentazione\Documentazione%20Mouse%20Mover.docx#_Toc185427970)

[Figura 10 18](file:///D:\Scuola\GitHub\ProgettoM306\3_Documentazione\Documentazione%20Mouse%20Mover.docx#_Toc185427971)

[Figura 11 20](#_Toc185427972)

# Allegati

* Diagrammi di flusso
* Requisiti
* Use case
* Gantt preventivo
* Gantt consuntivo