Indice

Capitolo 1

Analisi

Il seguente capitolo contiene l'analisi dei requisiti e del problema, ovvero fornisce una descrizione del dominio applicativo e delle funzionalità offerte dall'applicazione.

1.1 Requisiti

L'obiettivo del progetto è di realizzare un'applicazione di generazione di mappe di copertura di rete all'interno di un edificio, che simuli la qualità minima dei segnali a seconda delle infrastrutture presenti.

L'utente avrà a disposizione una mappa in cui indicherà la planimetria dei locali su cui effettuare la misura, il posizionamento di apparati Wi-Fi master/access points (generatori di segnali radio) e la locazione di apparati slave. Soddisfatto della rappresentazione, l'utente potrà vedere generata in una sezione dedicata una proiezione dell'intensità del segnale, calcolata con un misto di formule fisiche e procedure empiriche user-friendly.

Requisiti funzionali:

* L'applicazione dovrà presentare un'interfaccia utente completa di istruzioni accessibili, comprensibili e facili da applicare che indichino limiti e modalità di utilizzo del programma. Vi sarà un'area di disegno per la costruzione e l'editing del modello; l'utente potrà generare un risultato, che verrà visualizzato in un'area appropriata

* La planimetria dell'edificio comprenderà ostacoli architettonici con diversa capacità di assorbimento del segnale Wi-Fi, calcolata empiricamente

* piazzamento sulla mappa di generatori di segnali con potenza e banda precise e relativa generazione (sullo spazio di visualizzazione dei risultati) di mappa di distribuzione del segnale con indicazione dell'intensità tramite una scala di colori accessibile ai daltonici

* piazzamento sulla mappa di utilizzatori: punti fissi in cui è importante valutare il segnale

* cambiamento delle caratteristiche dei singoli elementi della mappa

* reset della mappa, cancellazione di singoli elementi non più ritenuti utili e disattivazione/attivazione di elementi (per evitare di doverli cancellare e reinserire)

* possibilità di lavorare su più mappe e duplicare una mappa su cui si sta già lavorando

Requisiti non funzionali:

* impostazione del set di colori a scelta dell'utente

* discriminazione effetti a seconda dell'antenna dell'access point (direzionale od omnidirezionale)

* rappresentazione dei dati user-friendly anche in seguito alla sovrimposizione di più aree

* accessibilità dei singoli elementi per modifica o cancellazione anche in caso di cluttering della mappa

* evitare interferenze indesiderate tra istanze di mappe

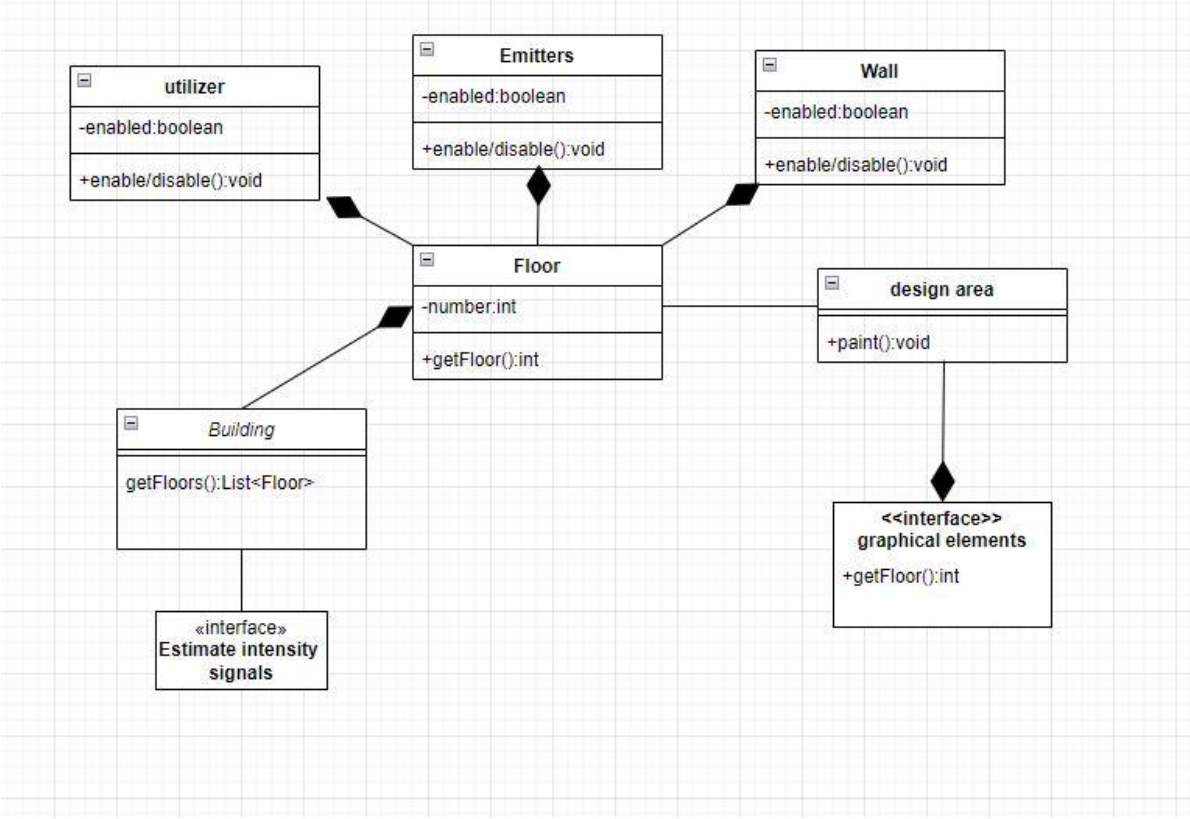
1.2 Analisi e modello del dominio

Partendo dal piano Terra, l'utente potrà creare i piani necessari a riprodurre accuratamente l'edificio da simulare. L'edificio può essere composto di piani differenti, ognuno dei quali avrà una relativa mappa per la memorizzazione dei dati ed una matrice per il risultato.

Inseriti i dati, l'utente potrà generare una stima dell'intensità dei segnali – la quale terrà in considerazione anche gli emittenti di piani distinti da quello corrente; tramite una checkBox, si potrà aumentare la precisione della stima.

È previsto un meccanismo per disabilitare gli elementi disegnati; non sarà necessario riattivarli per cancellarli definitivamente.

Ogni elemento possiede caratteristiche grafiche che ne rappresentino le proprietà principali per distinguerlo inequivocabilmente, coadiuvato da un riepilogo in dedicate caselle di testo



Design

2.1 Architettura

Il progetto utilizza il pattern architetturale MVC (Model-View-Controller): le componenti grafiche e logiche sono gestite separatamente; tale strategia si è rivelata fondamentale per il lavoro di gruppo, permettendo modifiche agevoli senza bisogno di continua interazione.

Il principale svantaggio di tale modello è che l'applicazione è condensata di una singola, corposa classe – mentre l'interfaccia è frammentata a seconda delle componenti da visualizzare; abbiamo saputo usare tali peculiarità a nostro vantaggio: le componenti essenziali dell'applicazione sono state programmate in maniera flessibile e adattiva, mentre le parti visibili dall'utente sono state affinate ai relativi scopi.

Il progetto contiene dieci classi custom:

- View
  - canvasPanel, che visualizza il risultato delle operazioni dell'utente
- Controller
  - FloorLogicImpl, che gestisce i piani in cui è diviso l'edificio
  - Floor, che gestisce gli elementi del Model in memoria – nonché i relativi simboli grafici
  - TutorialGUILogicImpl, che manipola i pulsanti e scorre le slide del tutorial
  - CanvasLogicImpl, immagazzina e gestisce le risorse grafiche
- Model
  - Emitters, che modella gli emettitori di segnale Wi-Fi
  - Walls, che struttura le barriere componenti la planimetria
  - Utilizers, che rappresenta gli apparati fruitori dei segnali degli Emitters
  - TutorialGUI, che crea una finestra d'aiuto più user-friendly dell'effettiva didascalia
  - CaptionGUI, immagazzina i dati per la view per la didascalia

L'applicazione stessa svolge compiti relativi alla view, complementando canvasPanel.

2.2 Design dettagliato

Alessandro Cacciaguerra

Il mio lavoro si è concentrato principalmente sulla gestione del controller: validazione (controlli di errore), salvataggio ed elaborazione dei dati (calcolo dell'intensità dei segnali). Ho creato inoltre la prima versione della rappresentazione grafica del piazzamento di emittenti, muri ed utenti e la modifica della scala di colori.

Emitters

Gestisce gli emittenti di segnali wireless; ognuno di questi ha una posizione nella planimetria, una determinata potenza (espressa in milliWatt) ed una relativa frequenza (espressa in MegaHertz). Tali valori non sono a scelta completamente libera dell'utente: affinché la simulazione presenti uno scenario realistico, i possibili valori sono ristretti in modo da adempiere alle normative ETSI EN (European Telecommunications Standards Institute – European Norm).

I suddetti campi sarebbero sufficienti per antenne omnidirezionali; per gestire antenne direzionali vi è una coppia di interi che rappresenta gli angoli di orientamento dell'antenna – indispensabili non solo

per determinare dove il segnale è distribuito ma anche per calcolare il “guadagno sul fronte” (una maggiorazione dell’intensità del segnale dovuta alla sua concentrazione, inversamente proporzionale all’ampiezza). Gli angoli avranno effetto anche sulla View: Emitters omnidirezionali verranno rappresentati come piccoli cerchi, mentre Emitters direzionali saranno disegnati come sezioni angolate verso la distribuzione del segnale. Per motivi grafici, si impone una distanza minima fra Emitters di 25 centimetri; considerando le dimensioni fisiche degli apparati Wi-Fi, non si ritiene il vincolo un fattore limitante.

Walls

Barriere presenti nell’edificio, non necessariamente effettivi muri. A differenza delle altre classi nel progetto, la posizione è una linea; sono differenziati in base all’impatto sull’intensità dei segnali, che l’utente può discernere in maniera empirica consultando la didascalia. Graficamente, un muro ad impatto più alto avrà una linea più spessa dalla colorazione più scura e viceversa.

Utilizers

Gli apparati slave che si servono dei segnali degli Emitters; si tratta di quadrati ben visibili (tratteggio spesso bianco e nero): non sono coinvolti nel calcolo dei segnali, ma nelle mappe di risultato vengono disegnati sempre in primo piano. Non si riferiscono al punto specificato (non sarebbero abbastanza evidenti) ma all’area di calcolo del segnale in cui il punto è presente.

Una volta creati gli elementi, le loro proprietà sono scritte in relativi JTextField (non modificabili dall’utente) in basso a destra – sei in totale, per separare elementi abilitati e disabilitati.

Per ognuno dei suddetti elementi, vi è una funzione per:

- creare (controlla il contenuto dei relativi JTextField e se non trova errori inserisce il contenuto nella mappa e nella casella di testo appropriate e chiama la funzione per dipingerlo nell’area di disegno)
- cancellare (se l’elemento è presente nella mappa del piano, lo rimuove dall’HashMap, dall’area di disegno e dalle caselle di testo senza compromettere gli altri elementi)
- abilitare/disabilitare (una cancellazione reversibile)

Per selezionare un campo da abilitare/disabilitare/cancellare è sufficiente indicarne la posizione e premere il relativo pulsante; sarebbe superfluo pretendere di avere il resto dei campi impostati correttamente in quanto la posizione è in ogni caso univoca.
Inizialmente si era pensato di permettere al pulsante abilita/disabilita di creare un nuovo elemento qualora i valori inseriti non fossero esistenti, ma si è deciso di rinunciare a tale funzionalità per evitare overlap con la funzione di creazione ed inserimenti accidentali da parte dell’utente.

Buona parte delle operazioni è eseguita dalla classe Floor, sviluppata insieme ad Andrea.

Floor

I tre suddetti elementi sono salvati in relative HashMap<(Elemento),Boolean> nel Floor; il “Boolean” servirà a discernere gli elementi abilitati (default) e disabilitati.

Floor ha funzioni per inserire/rimuovere gli elementi – fisicamente in memoria o graficamente nel relativo CanvasPanel

I metodi di manipolazione della memoria avrebbero potuto richiamare direttamente i metodi grafici - ma accorparli sarebbe stato scomodo e avrebbe introdotto rigidità.

Generazione risultato

Per generare il risultato, stabilisco le intensità relative ad ogni piano: se vi sono Emitters non disabilitati nel piano in questione, calcolo il valore dell'intensità per punti [ogni 50 centimetri, partendo da (25;25)]. Se un Emitters è direzionale, salto i punti non contenenti il relativo angolo e calcolo ed aggiungo il guadagno sul fronte agli altri. Deduco l'attenuazione del segnale causata dallo spazio e da eventuali muri non disabilitati intersecati. Sommo le intensità nel punto di tutti gli Emitters presenti e salvo il risultato in una matrice (intensity) che caricherò nel relativo Floor.

Infine, aggiungo alle intensità del piano corrente i risultati degli Emitters in altri piani dell'edificio (pavimenti e soffitti si presumono avere impatto alto) e coloro l'area in base al risultato ottenuto. A scelta dell'utente (mediante checkbox) posso scrivere i valori esatti dell'intensità (approssimati all'intero, in un colore contrastante) nelle aree dipinte – utile anche per i daltonici.

ValidateImpl

La classe riunisce tre piccoli metodi di utilità:

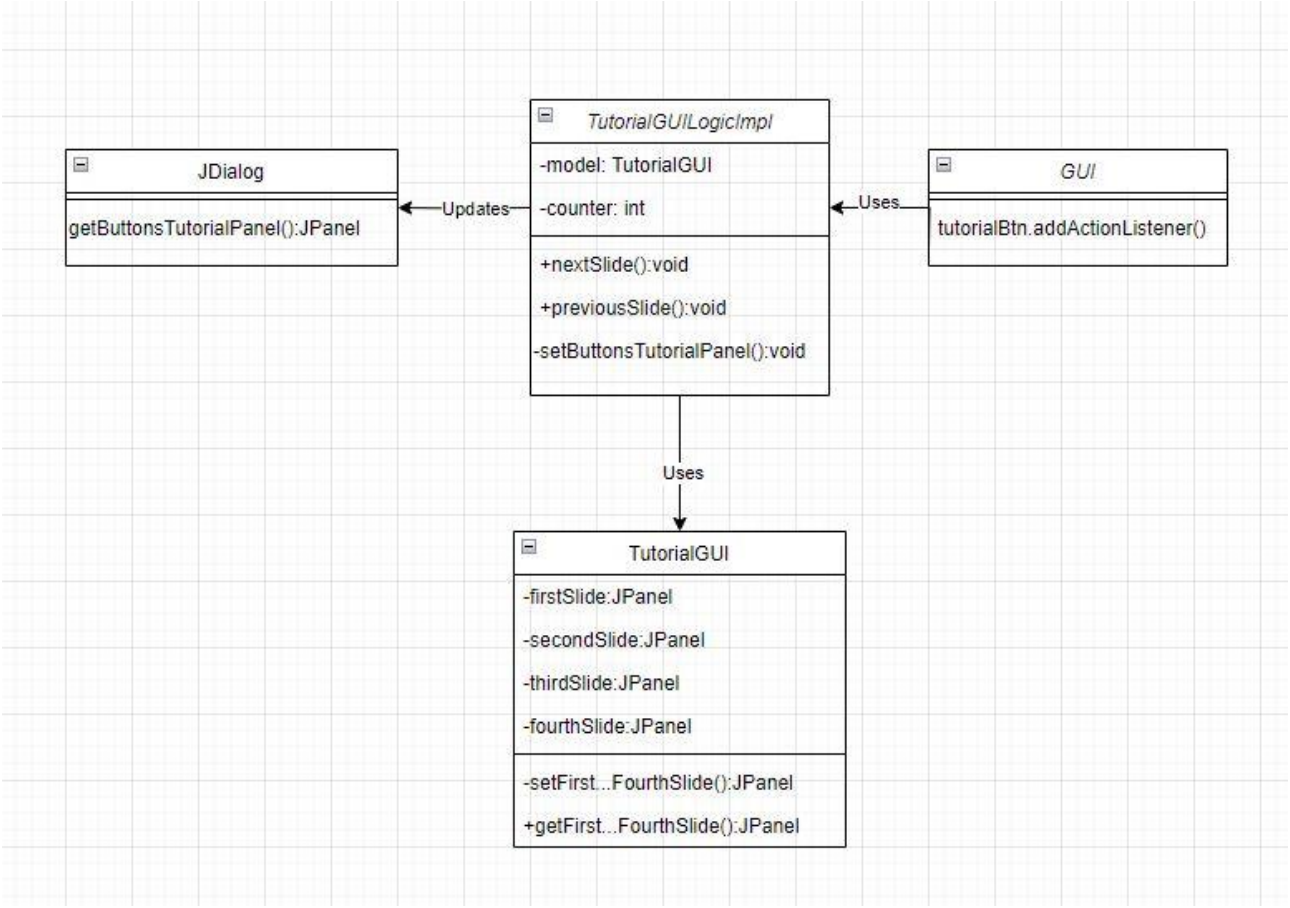
- boolean validatePosition(Point) valida la posizione di Emitters ed Utilizers;
- boolean validateWallPosition(Line2D) compie la stessa operazione, adattato alle specifiche di Wall;
- int getWallWidth(String) traduce l'impatto di un muro sul segnale nello spessore della relativa linea.

Altro

La Didascalia prevede un menù a tendina in cui selezionare un range di valori a cui associare un colore non già presente; il risultato prodotto sarà mostrato come la colorazione di un pulsante.

Andrea Casali

Inizialmente la mia parte era più rivolta alla gestione del View e del Model, occupandomi della creazione della gui, disegno delle risorse sul canvas e immagazzinamento e gestione dei dati per i controller usati dal mio compagno di gruppo, ovvero per esempio le classi Floor e FloorLogicImpl. Successivamente ho implementato altri elementi come TutorialGUI o CaptionGUI tenendo in considerazione l'uso del nostro programma da parte diun teorico utente casuale. Il pattern MVC è il pattern principalmente usato anche nella mia parte di progetto ma in alcuni casi, come Il CaptionGUI, la casse è separata per ridurre il carico della creazione di separate finestre e impedire possibili crash della GUI principale.



Sviluppo

3.1 Testing automatizzato

Alessandro Cacciaguerra

Nelle prime fasi di sviluppo, per collaudare il mio codice ho dovuto talvolta realizzare dei piccoli layout approssimativi – che Andrea avrebbe rimpiazzato con degli schemi perfezionati.

Per i test manuali abbiamo usato un edificio di 12 piani: 3 piani sotterranei, il piano Terra e 9 piani superiori – prima della creazione di FloorLogicImpl e relativi metodi, si referenziava il piano tramite un vettore e si sommava un offset costante a 3; se si raggiungeva un estremo i pulsanti erano disabilitati. Traducendo il codice, ho riutilizzato il principio dell'offset per correggere un bug che non distingueva tra piani inferiori e superiori.

Come prova finale, ho ricreato planimetria ed apparati Wi-Fi della mia abitazione per poi misurare l'effettiva potenza del segnale; l'errore medio ammontava a 12 dbm.

Andrea Casali

Nelle prime fasi, come specificato dal mio compagno, ho realizzato diversi test manuali usando diverse GUI minori che creavo di volta in volta. Successivamente ho continuato a usare test manuali in quanto ero in carico della grafica e quindi risultavano più conveniente per me specialmente data la piccola

dimensione del progetto, ma successivamente quando eravamo vicini alla fine del ho deciso di utilizzato alcuni test automatici, realizzati con Unit, per verificare il funzionamento delle parti model e controller di TutorialGUI, CaptionGUI e CanvasPanel.

3.2 Metodologia di lavoro

Alessandro Cacciaguerra

Per prima cosa ho creato degli snippets di generateResult(), per analizzare il problema. Ricevuto da Andrea uno scheletro di GUI.java, ho provveduto a popolarlo con le actionPerformed dei vari pulsanti ed un abbozzo della didascalia. Piuttosto che decidere le classi apriori, ho implementato il codice servendomi di strutture dati elementari e metodi disgiunti per poi accorparli in classi rifinite. Costruito un prototipo funzionante, Andrea ha strutturato un layout completo ed aggiunto funzionalità come scrolling, display di errori ed un pulsante già sviluppato per l'apertura di una nuova finestra. Io ho proceduto a creare le ultime classi e correggere errori/imperfezioni in quelle già sviluppate mentre Andrea ha apportato aggiustamenti e migliorie al layout.

Era mia intenzione usare Canvas e JavaFX, ma Andrea mi ha fornito un'alternativa migliore estendendo JFrame

Mi ero proposto di usare Pair per gli angoli di inizio e fine in Emitters, in quanto si trattava più di una coppia che di un Point; tuttavia, non volevo caricare un ulteriore import per una componente dati perfettamente funzionante – e non sarebbe stato accurato definirli un abbinamento chiave-valore.

Io ed Andrea abbiamo proceduto a staffetta, scambiandoci linee guida in modo da poter rimanere indipendenti e mettendoci d'accordo in caso di opinioni contrastanti; avevamo copie dei files sia in fork su GitHub che in locale, così da poter tornare indietro in caso di errore. Poiché le nostre componenti erano distinte, ognuno poteva aggiungere al proprio lavoro piccoli elementi non annunciati facilmente asportabili (es. visualizzazione degli errori, efficientamento dei controlli…) che non interferissero col lavoro dell'altro – purché li segnalasse sul momento.

Per gestire i piani, ho creato i meccanismi di Floor – che Andrea ha separato nelle classi Floor e FloorLogicImpl; similmente, ho scritto la caption prima che venisse inserita in una nuova finestra.

Essendovi metodi presenti fuori da una classe, li ho raggruppati in ValidateImpl ed ho trasferito su Floor le funzioni dei pulsanti della classe principale (stava diventando eccessivamente corposa ed eterogenea).

Andrea Casali

Il mio primo incarico era di creare la GUI, ovvero i pulsanti, textbox, textArea.., fornendo occasionalmente supporto al mio compagno, come ad esempio la creazione delle classi Emitters, Walls e Utilizers per rispettare il pattern MVC.

Successivamente ho continuato a creare Gui cercando di trovare un aspetto grafico che mi piacesse oltre a fornire una disposizione adatta per i dati e in quanto mi trovavo spesso a chiedere chiarimenti al mio compagno riguardo alcuni dettagli del progetto decisi di creare la didascalia Io mi ero occupato della creazione della JDialog che conteneva la didascalia mentre il mio compagno si era occupato di scrivere la descrizione oltre a decide di aggiungere il cambio il colore per le frequenze. A quel punto il progetto era arrivato alla parte in cui dovevamo disegnare gli elementi sul canvas, Alessandro aveva inizialmente gestito questo compito ma successivamente si era trovato in difficoltà in quanto il paint

degli elementi si cancellava ogni volta che si aggiornava il canvas, ci eravamo messi entrambi a trovare una soluzione e grazie a ScrollDemo, un esempio di programma che creava una situazione simile alla nostra trovato da alessandro, ho creato CanvasPanel. Abbiamo successivamente deciso che avrei gestito io canvasPanel in quanto rientrava nella parte grafica.

Verso la fine del progetto continuavo a fare debugging e succesivamente ho poi deciso,su consiglio della mia famiglia, di creare TutorialGUI per aiutare ipotetici utenti casuali a usare il programma senza il nostro aiuto.

3.3 Note di sviluppo

Alessandro Cacciaguerra

Gli Utilizers sono stati disegnati sovrapponendo due quadrati tratteggiati, il secondo con offset opportuno

validateWallPosition ha controllato la compenetrazione di segmenti tollerando un solo punto in comune

Ho fatto di ValidateImpl una classe statica, affinché i metodi fossero chiamabili anche da Floor

Andrea Casali

CanvasPanel è un extend di JPanel il quale fa un override del metodo paintComponent

In TutorialGUI per avere le parole di vari colori nel textPane uso il metodo appendToPane trovato su StackOverflow

Commenti finali

4.1 Autovalutazione

Alessandro Cacciaguerra

L'applicazione prodotta risponde alle funzionalità necessarie, ha uno scenario d'uso realistico e si può considerare user-friendly; il lavoro di gruppo è stato proficuo e soddisfacente: non ci sono state interferenze ed i lavori dell'uno sono stati accuratamente integrati dall'altro. Procedendo nella compilazione del progetto, abbiamo interagito per coordinarci con frequenza sempre maggiore – ma le iniziative sono state sempre ben accolte. Ciò affermato, il lavoro non si potrebbe definire di livello professionale in quanto i dati si affidano ad empirismo, approssimazioni e condizioni limitatorie che potrebbero mettere in discussione l'accuratezza dei risultati – pur mantenendo il margine di errore a livelli accettabili. Sono amante delle telecomunicazioni e mi ritengo soddisfatto di poter dedicare le mie competenze informatiche a tale obiettivo. In futuro, ho intenzione di inserirmi in un gruppo dedicato e produrre applicazioni similari.

Andrea Casali

L'applicazione prodotta mi risulta relativamente soddisfacente, nonostante penso che l'aspetto potrebbe essere un po' migliorato penso che il resto del programma funzioni efficacemente, magari non a un livello professionale ma comunque relativamente adeguato per un progetto.

4.2 Difficoltà incontrate

Alessandro Cacciaguerra

La mia maggiore difficoltà nello svolgimento del progetto è stata la scelta delle strutture dati da utilizzare nel programma: ho dovuto tornare indietro sulle mie scelte ogniqualvolta un certo tipo di dati non fosse compatibile con le nuove modifiche o ne avessi trovato uno più performante nelle nuove circostanze. Inoltre, dipendendo da Andrea per la componente grafica, ho dovuto crearmi dei layout posticci per il collaudo di nuovo codice da me implementato – che il compagno avrebbe rimpiazzato con strutture perfezionate.

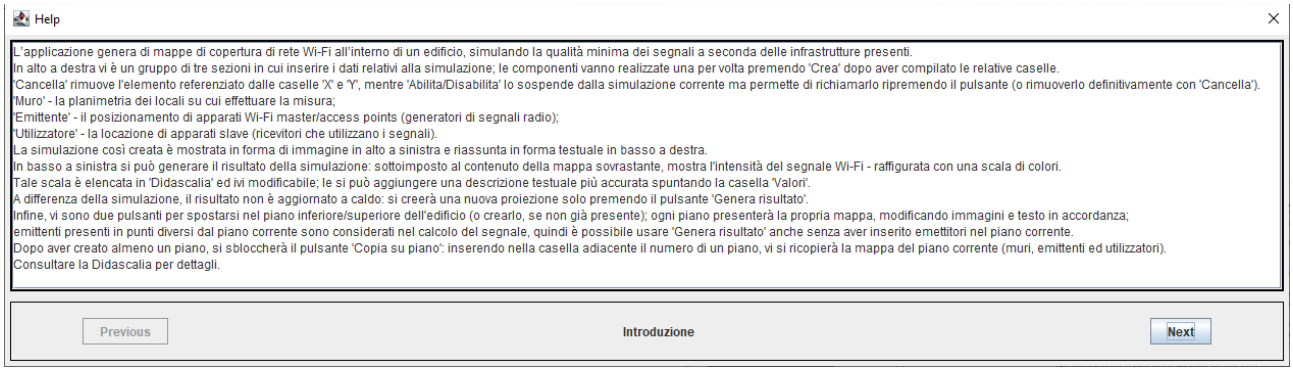
Andrea Casali

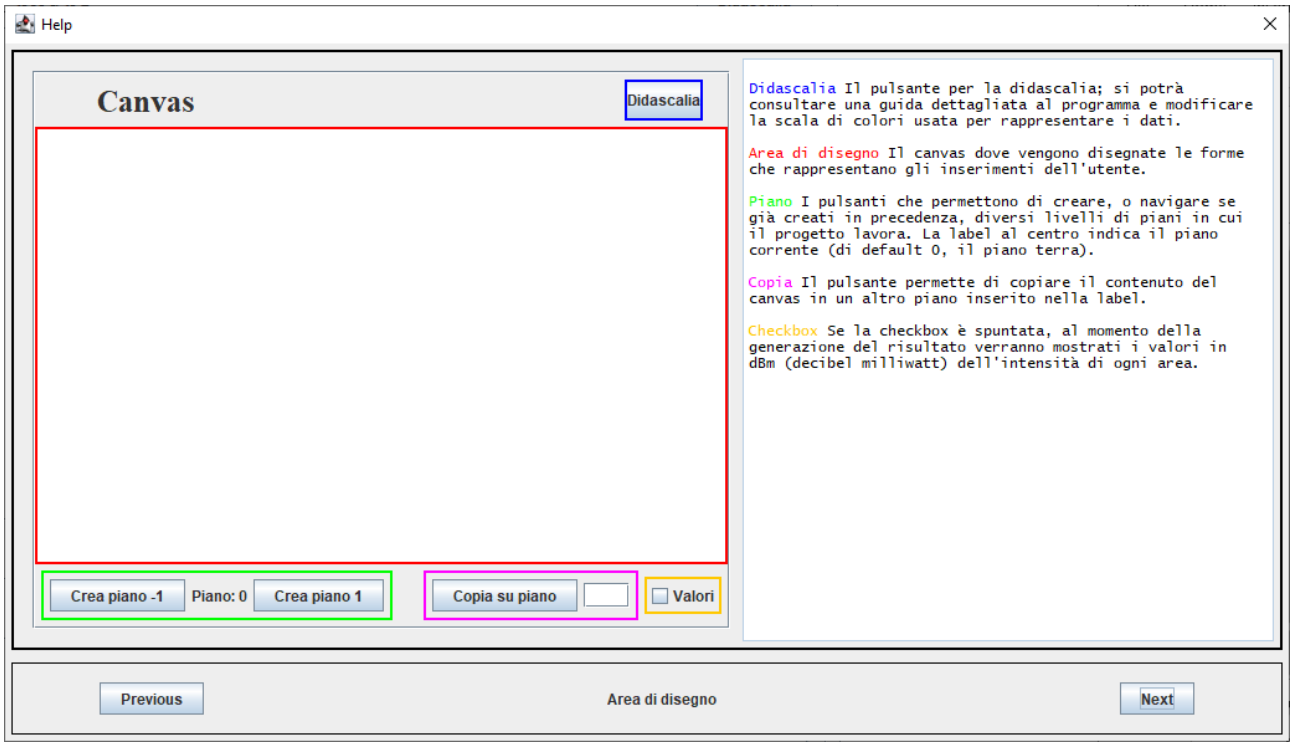
Le maggiori difficoltà che ho incontrato solo dovute al fatto che sono piuttosto distante dal normale periodo del corso, trovandosi così in difficoltà ad avere le conoscenze adeguate per poter supportare meglio il mio compagno di gruppo.

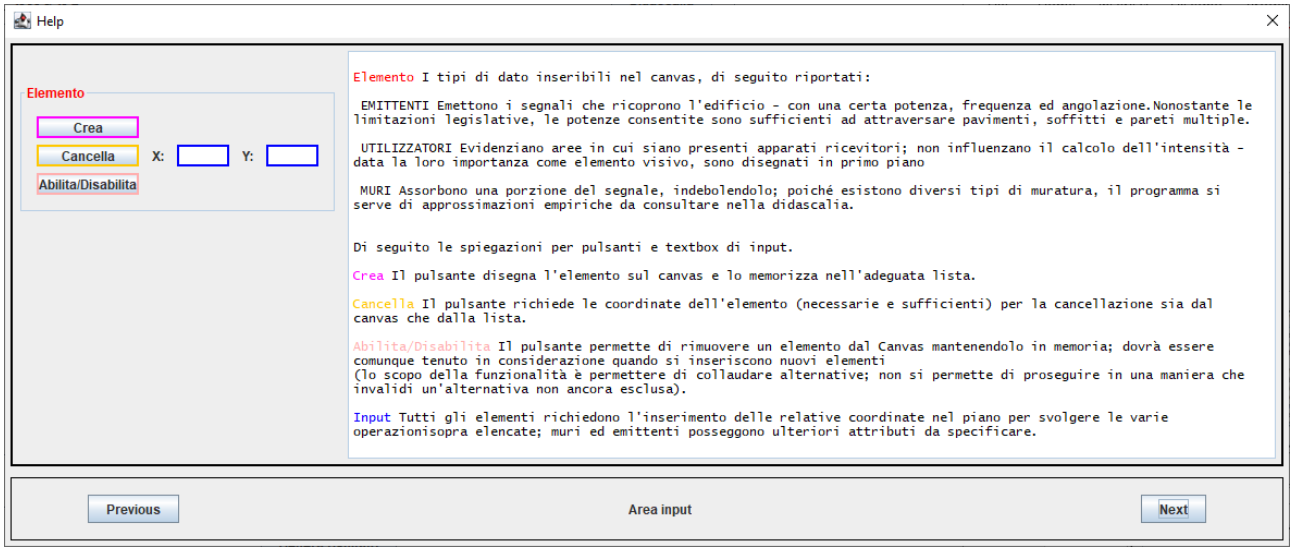
Appendice A

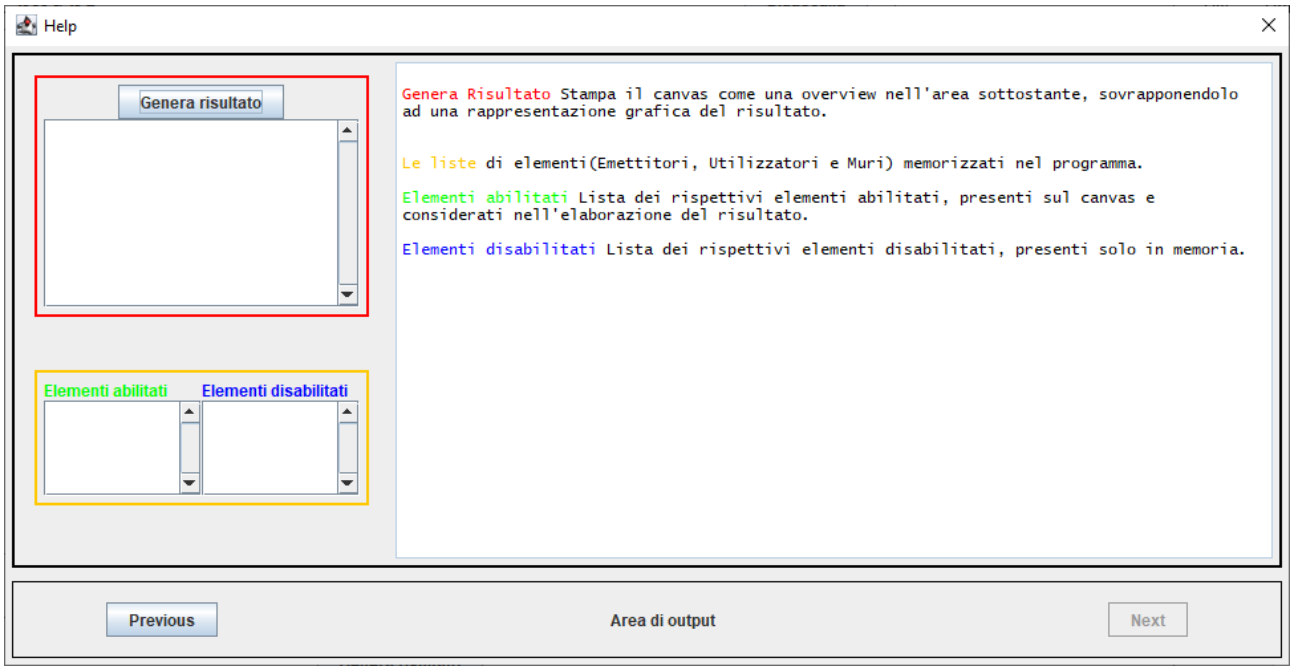
Guida utente

Screenshots di HelpDialogImpl









Copia della didascalia

Effetto del materiale dell'ostacolo su un segnale radio

A seconda del materiale, gli ostacoli possono riflettere le onde radio, assorbirle, privandole di una parte della potenza, o non avere alcun effetto sul segnale radio. Tali materiali sono chiamati radiotrasparenti. Più alto è il coefficiente di assorbimento del segnale e più spesso è l'ostacolo, più forte è l'impatto sulla trasmissione radio.

Coefficiente di assorbimento del segnale

Basso

Perdita di potenza del 50%

- Mattone rosso secco di 90 mm di spessore

- Pannello di gesso di 100 mm di spessore

- Legno secco di 80 mm di spessore

- Vetro di 15 mm di spessore

Medio

La potenza si riduce di 10 volte

- Mattone di 250 mm di spessore

- Blocchi di calcestruzzo di 200 mm di spessore

- Calcestruzzo di 100 mm di spessore

- Muratura di 200 mm di spessore

Alto

La potenza si riduce di 100 volte

- Calcestruzzo di 300 mm di spessore

- Calcestruzzo armato di 200 mm di spessore

- Travi in alluminio e acciaio

Di default, l'edificio è composto dal solo piano Terra, ma si possono aggiungere piani superiori o sotterranei

Su ogni piano, si provvede uno spazio di 10 metri per 8 in cui riprodurre la pianta dell'edificio; valori illegali saranno considerati 0.

Porte, finestre interne ed esterne od altre aperture fra stanze sono sempre considerate chiuse ai fini della rilevazione del segnale: si ipotizza il segnale minimo nel caso peggiore.

La precisione massima nel piazzamento di un muro è di 50 centimetri, sugli assi x e y; non si accettano muri diagonali.

Piani strutturati inframmezzati da piani vuoti od incompleti sono tollerati: si presume che le relative planimetrie siano ininfluenti ai fini della simulazione dell'utente.

Per selezionare un campo da abilitare/disabilitare/cancellare è sufficiente indicarne la posizione e premere il relativo pulsante; non serve impostare correttamente il resto dei campi.

Secondo le normative ETSI EN, la potenza di emittenti wireless in un edificio non può superare i 200 milliWatt e la frequenza deve essere compresa nella banda 5150-5350 MegaHertz.

Gli emittenti devono essere distanziati di almeno 25 centimetri da altri emittenti e di almeno 1 centimetro da potenziali muri (le coordinate non possono essere divisibili per 50).

I muri si possono intersecare ma non compenetrare.

Gli angoli di inizio e di fine sono 0 e 360 per antenne omnidirezionali; per antenne direzionali, l'ampiezza è calcolata in senso orario con 0° = 360° = ore 3.

Si presume che tutti gli emittenti, direzionali ed omnidirezionali, siano ottimizzati per trasmettere segnali nel piano in cui sono collocati: i segnali propagati verticalmente sono considerati diffusione.

Il guadagno rispetto ad un'emittente isotropica si ritiene già incluso nella potenza; se così non fosse, triplicare quanto si intendeva inserire