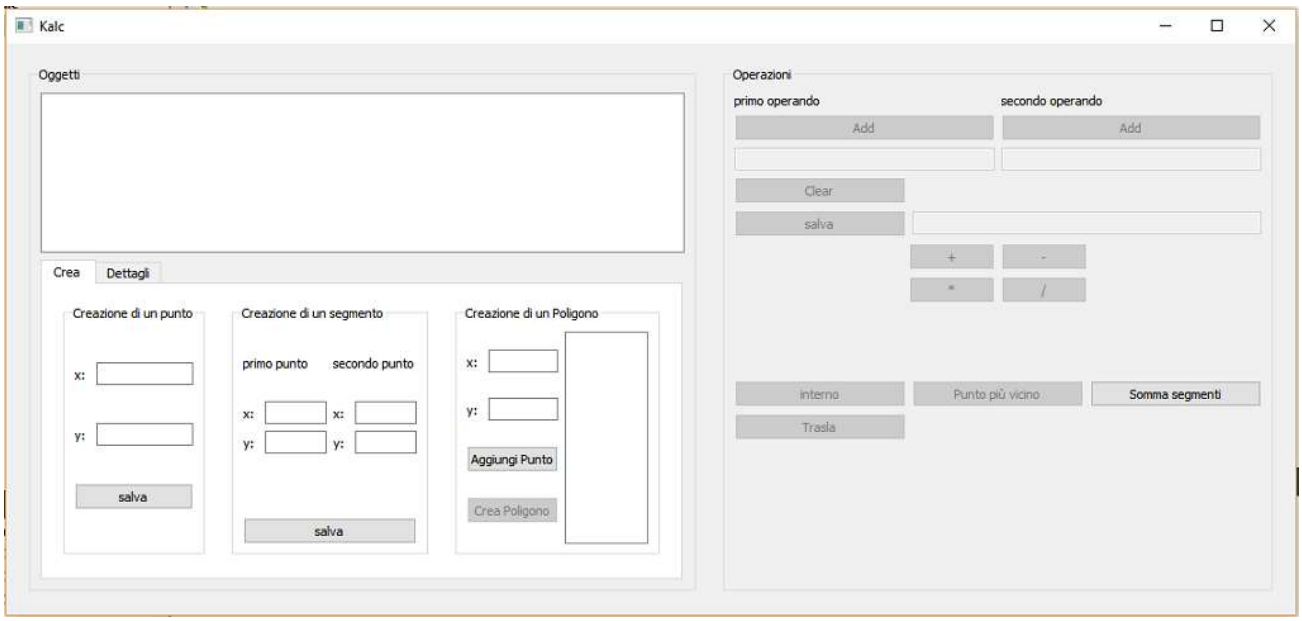
Bratzu Gian Marco
Matricola: 1049631
Anno accademico 2017-2018

KALC

Progetto di programmazione ad oggetti



Sistema Operativo di Sviluppo: Windows 10
Versione Qt: Qt 5.9.1 Versione
Qt Creator: 4.5.0 Versione
Qt Laboratorio: 5.5.1 Versione
QMake Lab: 3.0

Bratzu Gian Marco
Matricola: 1049631
Anno accademico 2017-2018

Introduzione:

Lo scopo del progetto è quello di fornire un'applicazione, completa di interfaccia grafica, che permetta di svolgere operazioni fra almeno tre diversi tipi di oggetti. Nel caso della calcolatrice grafica Kalc questi sono: punti, segmenti e poligoni.

Descrizione del modello:

- **Object** è la classe base astratta da cui derivano tutte le altre classi.
  È formata da 3 metodi virtuali puri che vengono implementati in tutte le sottoclassi;

- la classe **Punto** è una classe concreta, caratterizzata dalle coordinate x e y che permettono di rappresentare un punto nel piano. Rende disponibili tutte le funzionalità di base per la gestione dei punti nello spazio oltre a due metodi statici, che permettono il calcolo della distanza tra due punti e l'angolo formato da 3 punti. La classe implementa un ordine totale che privilegia la coordinata x rispetto alla y.

- La classe **Segmento** è analoga alla classe Punto e rappresenta un segmento grazie alle coordinate dei due estremi.

  Nello specifico in questa classe è stata definita una funzione, che permette di calcolare il punto appartenente al segmento con distanza minima dal punto stesso. Tutto questo però non nella versione di Tolomeo, cioè la distanza di un punto dal punto medio di un segmento, bensì una versione ridefinita in cui dato un segmento AB consideriamo la striscia di piano definita dalle rette ortogonali al segmento e passante per i suoi estremi. Il punto O può trovarsi nel semipiano a sinistra o a destra della striscia o dentro la striscia stessa. Se O è posto dentro la striscia di piano la sua distanza da AB è la misura del segmento di perpendicolare OH portata da O ad AB, altrimenti è la misura del segmento che unisce O all'estremo di AB a lui più vicino.

  Anche qui è stato rispettato lo stesso ordine totale della classe Punto, dunque un segmento può essere contemporaneamente < e > di un altro.

Bratzu Gian Marco
Matricola: 1049631
Anno accademico 2017-2018

- La classe **Base** è una classe contenitore di Punti, che implementa alcune delle principali funzionalità di un contenitore quali add() e remove().

  Questa classe è necessaria per la definizione di Poligono tramite la funzione convex_hull() che ritorna un sottoinsieme di Punti che andranno a formare l'involucro convesso di un Poligono.

- La classe **Poligono** derivata dalla classe Base definisce un Poligono, convesso semplice, tramite l'ordine antiorario dei sui vertici.

  Utilizza la superclasse come contenitore di punti, mentre il campo dati di Poligono è una lista di puntatori agli elementi contenuti nel sottoinsieme dei punti restituito dalla funzione convex_hull() permettendo così di evitare lo spreco di memoria dovuto alla duplicazione dei punti.

  Nonostante Poligono sia definita come una lista di puntatori, la sua distruzione non provoca memory Leack, in quanto viene distrutto anche il suo sottoggetto che non sono altro che gli oggetti puntati dalla lista dei puntatori di Poligono.

  Tra i vari metodi relativi alle proprietà geometriche di un Poligono che sono area, perimetro ed equilatero, e i metodi interno e punto più vicino, vengono ridefinite le operazioni matematiche di somma e differenza, prima come inclusione o esclusione nella lista e solo dopo come differenza effettiva dei punti nello spazio. Per quanto riguarda la moltiplicazione e la divisione queste vengono definite nello spazio.

Descrizione della parte grafica:

La pare grafica si divide in due parti:

- **LeftArea**, la parte sinistra, gestisce la creazione e la visualizzazione degli oggetti. Permette la creazione dei 3 tipi di oggetto utilizzati in Kalc (punti, segmenti e poligoni) e la visualizzazione tramite selezione nella ListArea modifica la View in base al tipo selezionato.

- **RightArea**, la parte destra, opera sul tipo di calcolo scelto tramite la selezione e l'aggiunta degli operandi, che vanno a modificare lo stato della calcolatrice sbloccando i button alternativamente riferiti a operazioni unarie o binarie.

Il controller

La classe **controller** svolge il ruolo di intermediario tra il model e la view ed è implementato come una lista di puntatori alla classe Object.

Viene mantenuto in memoria durante tutta l'esecuzione del programma e il suo puntatore viene passato tra le varie classi per permettere di riferirsi tramite indice ai vari oggetti creati rendendo così netta la separazione tra view e model.

Nel controller vengono sviluppate tutte le funzioni polimorfe, in quanto tutti gli oggetti creati sono puntatori alla classe Object e di conseguenza devono essere castati per poter essere utilizzati.

Bratzu Gian Marco
Matricola: 1049631
Anno accademico 2017-2018

Il metodo Trasla e tutti i vari metodi di stampa utilizzano metodi virtuali puri su puntatori ad oggetti della classe astratta, rendendo il polimorfismo l'aspetto cruciale di questa classe.

Il metodo newObject() richiama il costruttore "giusto" in base al tipo puntato dall'oggetto salvato correntemente nella variabile risultato (res).

I metodi Somma, Differenza, Moltiplicazione, Divisione, Interno e Somma segementi effettuano dei dynamic_cast per permettere la chiamata polimorfa ai vari operatori definiti nel modello.

La gestione degli errori:

Tutti gli errori sono gestiti dalla classe **error**, la quale in base alla tipologia di errore genera dei QmessageBox di errore specifici.

Le tipologie di errore sono causate dalla divisione per zero, dal coefficente angolare di un segmento verticale e al tentativo di creare un poligono con meno di 3 punti o con tutti punti allineati.

Si è preferito non gestire i campi dati nella creazione degli oggetti, in quanto questi convertono qualsiasi dato immesso differente da un numero in 0.

Istruzioni di compilazione:

La cartella principale del progetto contiene le seguenti directory:

- **Java**: contiene la gerarchia in linguaggio Java.

- **Kalc**: contiene tutto il codice di Qt, compreso il file .pro che quindi non deve essere rigenerato e la cartella model contenente la gerarchia in linguaggio C++.

Per compilare correttamente la GUI, è necessario eseguire i seguenti dalla directory principale del progetto: ~$ cd Kalc ~$ qmake ~$ make

Per eseguire correttamente il programma, digitare da terminale: ~$ ./Kalc

Per compilare correttamente la parte java, è necessario eseguire i seguenti dalla directory principale del progetto: ~$ cd Java ~$ javac *.java Per eseguire correttamente il programma, digitare da terminale: ~$ java Use

Bratzu Gian Marco
Matricola: 1049631
Anno accademico 2017-2018

Tempo richiesto

- Analisi preliminare del problema 6~

- progettazione modello e GUI ~32h

- Apprendimento libreria Qt ~6h

- Testing e debugging ~4h

-Conversione del modello in Java ~2h