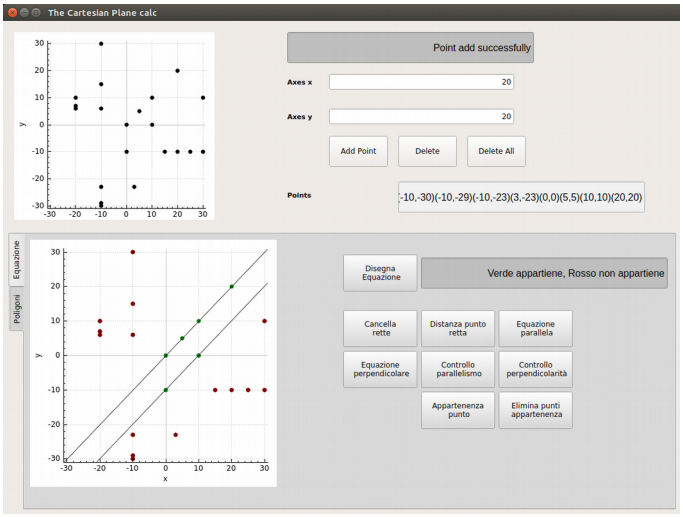
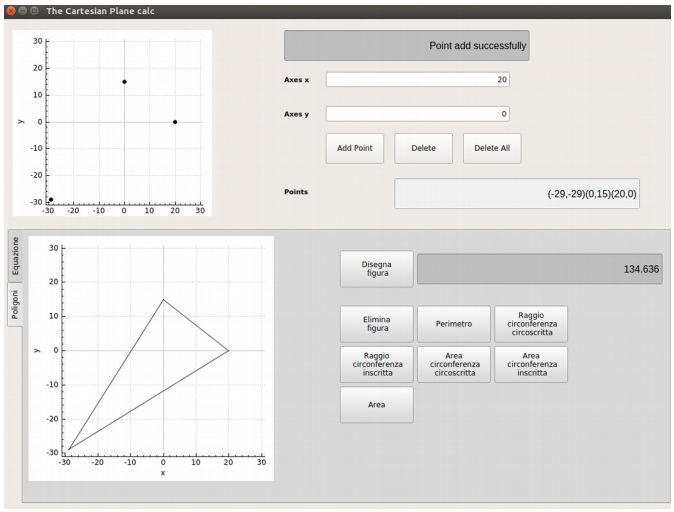
PROGETTO PROGRAMMAZIONE AD OGGETTI

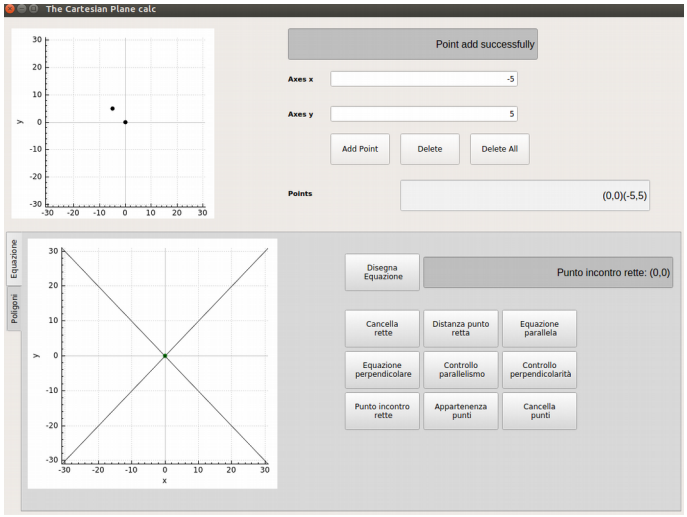
Anno 2017/2018

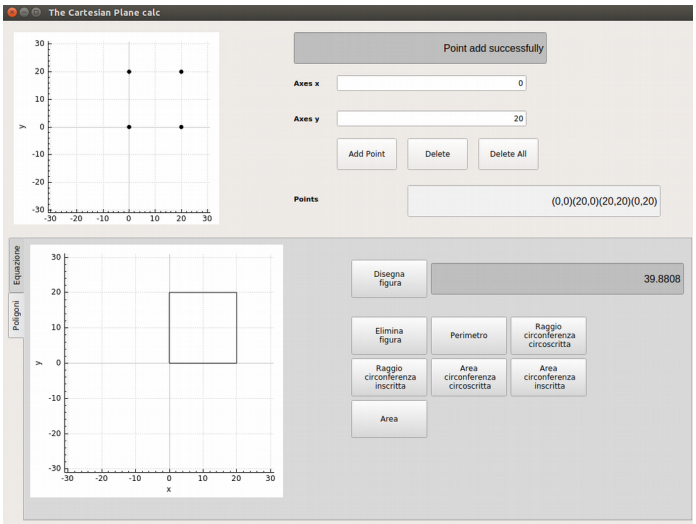
FRANCESCO CORTI 1142525

La calcolatrice nel piano cartesiano









Introduzione alla calcolatrice

La calcolatrice permette di effettuare conti ed operazioni su **figure presenti nel piano cartesiano** che verranno disegnate dall'utente tramite l'inserimento dei punti. Le operazioni effettuabili si dividono in due categorie, quelle per i poligoni regolari(triangolo o quadrato) e quelle per le rette. Le operazioni effettuabili per le **rette** sono:

- **Distanza del punto dalla retta:** $$d = \left| \frac{ax_0 + by_0 + c}{\sqrt{a^2 + b^2}} \right|$$
- **Controllo perpendicolarità di due rette:** $$m_1 = -\frac{1}{m_2}$$
- **Controllo parallelismo di due rette:** $m_r = m_s$
- **Retta perpendicolare passante per un punto:** $y - y_1 = -1/m_1(x - x_1)$
- **Retta parallela passante per un punto:** $$y - y_0 = m(x - x_0)$$
- Verificare l'**appartenenza di uno o più punti alla retta/e**

- **Punto di incontro tra due rette**

Le operazioni effettuabili per le **figure geometriche** triangolo e quadrato sono:

- **Area**
- **Perimetro**
- **Area circonferenza circoscritta**
- **Area circonferenza inscritta**
- **Raggio circonferenza inscritta**
- **Raggio circonferenza circoscritta**

Informazioni sullo sviluppo

Il progetto è stato sviluppato sul mio personal computeer avente Ubuntu 16.04 nell'ambiente Qt Creator versione 4.0.2 e versione del compilatore: gcc 5.4.0.

Informazioni sulla compilazione:

La compilazione del sorgente potrebbe presentare dei problemi a cause dell'utilizzo della libreria grafica QCustomPlot utilizzata per gestire le figure e i punti nel piano cartesiano. Si allega pertanto la seguente lista di comanda da aggiungere al file.pro sotto al campo INCLUDEPATH += affinché la compilazione avvenga correttamente:

CONFIG += c++11

QT+= core gui

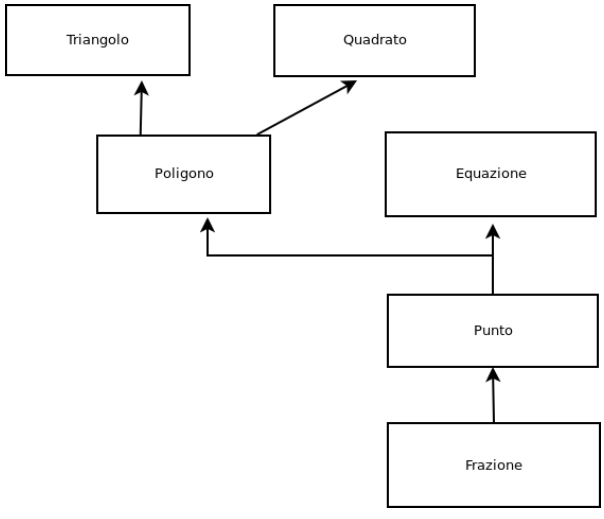
QT += widgets

QT += widgets printsupport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

In particolare spesso il fatto di usare c++11 mi ha dato dei problemi nella compilazione.

Oggetti resi disponibili dalla Gerarchia:



- **Frazione**(numeratore,denominatore):
  Oggetto che ha la funzione di rappresentare dei razionali attraverso due interi.

- **Punto**(Razionale x, Razionale y):
  L'oggetto Punto rappresenta un generico punto disegnabile nel piano piano cartesiano a due dimensioni.
- **Equazione**(Razionale y, Razionale x, Razionale q):
  L'oggetto Equazione rappresenta una generica equazione nel piano cartesiano.
- **Poligono**:
  Classe astratta della mia gerarchie dei poligoni regolari.
- **Triangolo**:
  L'oggetto triangolo rappresenta un generico triangolo nel piano Cartesiano è rappresentato da tre punti.
- **Quadrato**:
  L'oggetto quadrato rappresenta un generico quadrato nel piano Cartesiano ed è rappresentato da quattro punti.

La classe base astratta **poligono** presente i seguenti **metodi virtuali puri**:

- *Area Circonferenza Inscritta*
- *Area Circonferenza Circoscritta*
- *Perimetro*
- *Area*
- *Raggio Circonferenza Inscritta*
- *Raggio Circonferenza Circoscritta*

Ognuno di questi metodo è applicabile solamente grazie al metodo **distanza** presente nella classe punto che calcola la distanza tra i due punti presenti nel piano cartesiano e permette di calcolare quindi la lunghezza di tutti i lati del poligono.
Tutti i metodi virtuali puri vengono implementati nelle classi Triangolo e Quadrato.

La classe concreta **equazione** presenta i seguenti metodi:

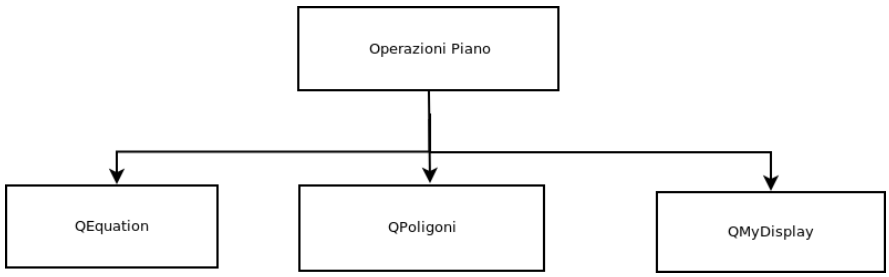
- *Incontro:* date due rette ritorna il punto di intersezione.
- *Distanza:* dato un punto non appartenente alla retta ritorna la distanza di quel punto dalla retta.
- *Perpendicolare*: data una retta e un punto ritorna la retta perpendicolare passante per quel punto.
- *Parallela:* data una retta e un punto ritorna la retta parallela alla retta considerata passante per quel punto.

Design della GUI:

La GUIè stata progettata con un attenzione particolare relativamente all'**estensibilità**, pertanto sono state definite due gerarchie che costituiscono totalmente la parte grafica della GUI.

Model:

Il model è composta da questa gerarchia:



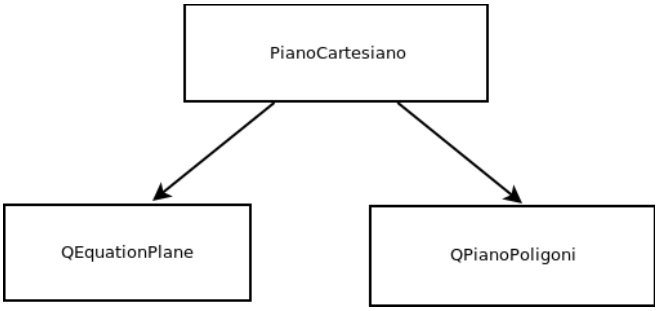
e prevede i seguenti **metodi virtuali puri** presenti nella classe base astratta Operazioni Piano :

- **setDisplayText**: mostra a schermo una stringa.
- **drawFigure**: data una lista di punti disegna una figura.
- **getNumeroFigure**: torna il numero delle figure presenti nel piano cartesiano.

Grazie all'utilizzo di questi metodi virtuali posso effettuare **chiamate polimorfe** dopo aver effettuato delle operazioni nel mio controller nelle figure o punti presenti nel mio display, ogni classe presente nel mio model è una classe derivata da QWidget.

View:

La mia view per rappresentare i dati immessi dall'utente è rappresentata dalla seguente gerarchia:



La classe base della view PianoCartesiano prevede **due metodi virtuali puri:**

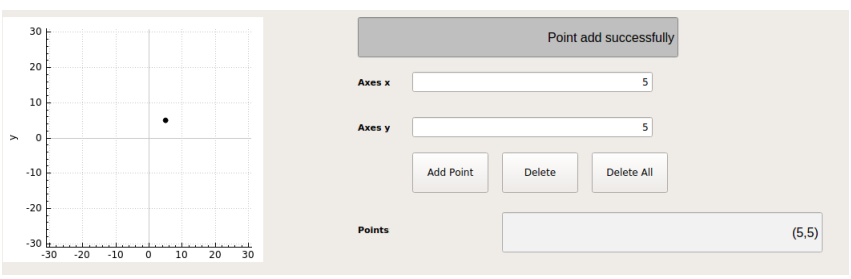
- **drawFigure**: disegna la\e figura\e nel piano cartesiano.
- **delFigure**: cancella le\a presenti nel piano cartesiano.

La view è composta da due classi **QEquationPlane** e **QpianoPoligoni** che sono classi derivate dalla libreria QCustomPlot.

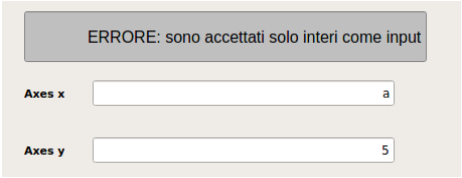
Per la gestione dei calcoli ho utilizzato la classe QMainCalc dove ho implementato dei metodi che gestiscono le operazioni da applicare sui dati immessi dall'utente. Tramite l'utilizzo della classe base astratta OperazioniPiano sono riuscito a rendere la gestione dei signal molto semplice sfruttando soprattuto chiamate polimorfe.

Manuale della GUI:

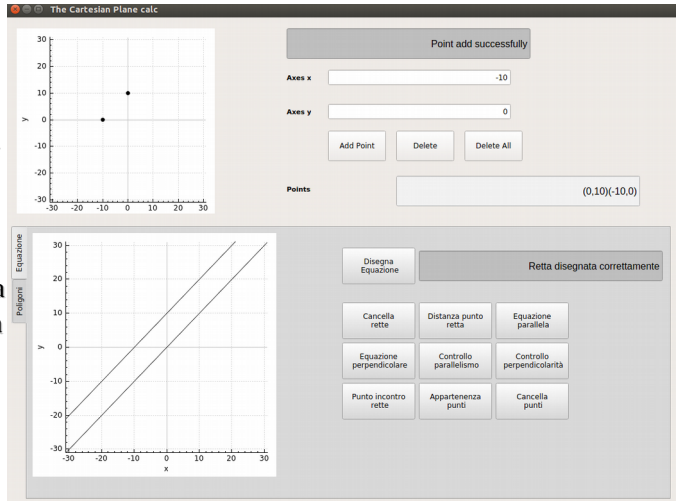
La gui presenta delle istruzione ben precise per il funzionamento.
L'utente deve inserire all'interno della riga Asse x e all'interna della riga Asse y due numeri interi compresi tra 0 e 30 (compreso) e schiacciare sul tasto Add point.



Se la digitazione avviene correttamente il punto verrà immesso nel display e verrà aggiunto nello schermo Points il valore del punto immesso. Attenzione che se per caso venisse digitato un valore scorretto ad esempio numero contenente una lettera o troppo grande verrà dato un segnale d'errore nel display. Es:



La **cancellazione dei punti** può essere effettuate tramite il tasto **Delete** che cancella il punto se c'è, con i valori che sono stati immessi dall'utente oppure tramite il tasto **Delete all** che dopo aver fatto apparire una finestra permette il cancellamento di tutti i punti presenti nel piano.

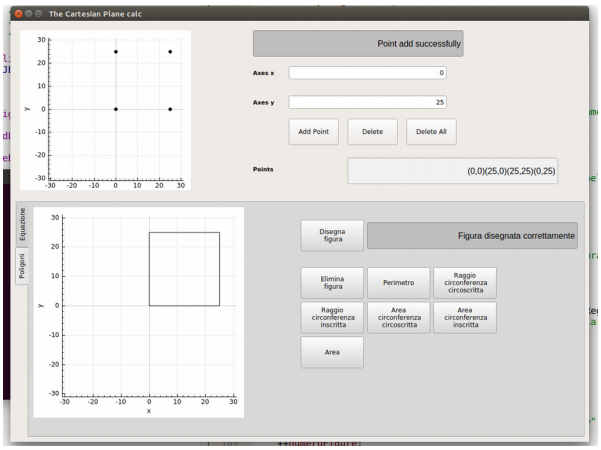


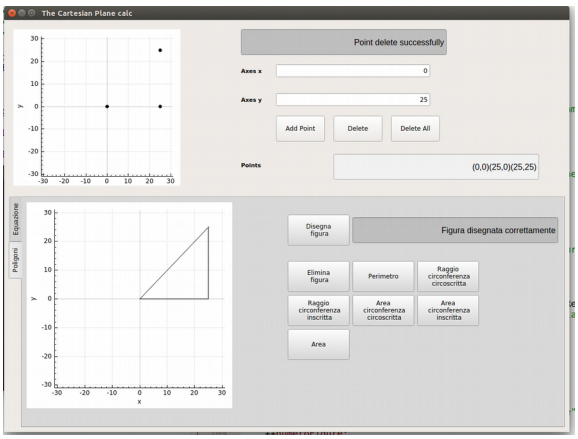
Per la **gestione delle rette** quando saranno presenti due punti nel piano sarà possibile creare una retta tramite il tasto Disegna Equazione se l'operazione andrà a buon fine comparirà sul display la scritta: "operazione avvenuta con successo". Ovviamente sarà possibile disegnare la seconda retta solamente se sarà una retta diversa dalla retta di partenza. Ogni operazioni è effettuabile secondo delle condizione:

- **Punto di incontro rette**: le rette devono essere due e non parallele.
- **Appartenenza punti**: una o due rette e dei punti nel display.
- **Equazione perpendicolare**: deve essere presente una retta e un punto nel piano.
- **Equazione parallela**: deve essere presente una retta e un punto non appartenente alla retta.
- **Controllo Perpendicolarità:** devono essere presenti due rette nel piano cartesiano.
- **Controllo Parallelismo**: devono essere presenti due rette nel piano cartesiano.
- **Distanza punto retta**: deve essere presente un punto e una retta nel piano.

Il tasto **Cancella Punti** permette di cancellare i punti disegnati dopo aver schiacciato il tasto appartenenza punti, se ce ne sono.
Il tasto **Cancella rette** permette di cancellare le rette che sono state disegnate nel piano, se ce ne sono.

Per la **gestione dei poligoni** la gestione dell'immissione dei punti avviene come descritto in precedenza per le rette, l'unica differenza è che i **quadrati** gestiti dalla mia calcolatrice sono solamente quelli regolari quindi viene fatto un controllo sulla distanza dei punti immessi e se risultano diverse viene fatto comparire un messaggio d'errore. Per le gestione dei **triangoli** non vi è nessuna restrizione sulla regolarità dei lati. Es:





Tutte le **operazioni** messe a disposizione per i poligoni sono effettuabili solamente in presenza del poligono nel piano, in assenza verrà visualizzato sul display un messaggio d'errore.

Polimorfismo:

Il polimorfismo è stato ampiamente utilizzato all'interno delle mia gerarchie di calcolo come poligoni dove è stato marcato ogni metodo della classe base virtual, ciò permette un' estendibilità della mia calcolatrice che per rimanere all'interno delle 50 ore non sono riuscito riuscito a trattare. Ho utilizzato molto il polimorfismo anche all'interno della GUI dove sono riuscito a far derivare le mia classi da delle classi basi astratte riuscendo così a compiere quasi tutte le operazioni principali attraverso delle chiamate virtuali.

Per ultimo la **tabella** con le ore impiegate per lo sviluppo del progetto:

Fase	**Ore**
Ore totali	70
Progettazione modello:	15
Analisi preliminare:	8
Progettazione GUI:	5
Sviluppo GUI:	20
Parte java:	5
Debug:	5