**Plot-Project, Day 3**  
------------------------------------

**Task 1: Make sure you have points added to your scene and that they emit signals**  
---------------------------------------------

1) Take a look at the Point header file. Add the following code to this file defining a custom signal and adding the mousePressEvent function.

Point.h

signals:  
 // for distance  
 void PointSelected(int x, int y);  
  
protected:  
 void mousePressEvent(QGraphicsSceneMouseEvent \*event) override;

Point.cpp

void Point::mousePressEvent(QGraphicsSceneMouseEvent \*event)  
{  
 qDebug() << "point clicked!";  
}

What parameters does the PointSelected signal take?

Int x, int y

**What line of code would you write to emit this signal from the mousePressEvent method in Point?**

2) Run your application. Has anything changed in terms of interaction? Why or why not?

When I click on a point now I shows that I did and I know that has been registered.

3) When you add a Point to your scene in your PlotWindow, connect that Point and it's PointSelected signal to a new slot in your PlotWindow class. This new slot must have the same parameters as the signal that it is receiving. Implement this slot so that it prints the x and y coordinates of the clicked point via qDebug().

Hint: your new slot needs to be a custom slot, like SlotTest from the previous lecture.

4) Will signals ever have return values? why yes/ why no?

**No, they are void as they just are activated upon click and do not pass on**

**If you missed plot project day 2, download the following zip.**

[ploproject\_day3.zip](https://canvas.colorado.edu/courses/99583/files/71856208?wrap=1)[Download ploproject\_day3.zip](https://canvas.colorado.edu/courses/99583/files/71856208/download?download_frd=1)

**Task 2: Making our Point change colors when we click it**  
**-----------------------------------------------**  
1) Finish implementing the void Point::mousePressEvent(QGraphicsSceneMouseEvent \*event) method in point.cpp by making it so that when a user clicks a point, it changes colors. This method is automatically called whenever a user clicks a Point. Copy + paste the code that you added here.

Hint: you may need to call the update() method in mousePressEvent to let the object know that it has been updated and needs to be re-painted.

2) What are the limitations on the parameters of slots/signals in Qt?

**Task 3: displaying distance**  
---------------------------------------------  
0) Change your PointSelected signal in Point to have the signature:  
void PointSelected(Point \* p). Change the slot that reacts in PlotWindow so that it matches. We're doing this to make calculating distance easier later.

Here is the code that we added/changed:

point.h

void PointSelected(Point \* p);

point.cpp:

emit PointSelected(this);

plotwindow.h:

void PointSelectedSlot(Point \* p);

plotwindow.cpp:

void PlotWindow::PointSelectedSlot(Point \* p) { qDebug() << p->get\_x() << ":" << p->get\_y();}

1) Make it so that the PointSelected signal is only sent when a point is right-clicked.  
You can access information about the mouse button clicked via the event->button() method. To see if a certain modifier was used, you can test if the flags returned == Qt::ModifierYouAreLookingFor.

Take a look at the following documentation:  
[https://doc.qt.io/qt-5/qt.html#MouseButton-enumLinks to an external site.](https://doc.qt.io/qt-5/qt.html#MouseButton-enum)

Copy + paste your Point::mousePressEvent method here.

2) Edit your slot in PlotWindow that reacts to the PointSelected signal so that when the user right-clicks a point, its information is filled in under the Point 1 label.  
Hints:  
- labels have a setText method that takes a QString as a parameter. A QString can take a c string in its constructor.  
Example:

std::string s = "Point 1: (" + std::to\_string(p->get\_x()) + ", " + std::to\_string(p->get\_y()) + ")";  
QString qs(s.c\_str());  
ui->point1Label->setText(qs);

3) Make it so when the user right-clicks a second point, the Point 2 label and the Distance labels are also updated.  
Hints:  
- you'll need to "save" information across calls to your slot in your PlotWindow—think about how adding field(s) to your PlotWindow can help you here.  
- Use the Point's Distance() method!

**Task 4: Adding a boolean delete\_mode\_ field to your project**  
-----------------------------------------------  
1) Add a boolean delete\_mode\_ field to your PlotWindow class. Don't forget to initialize this field in your PlotWindow constructor file and add any appropriate getters/setters! Did you make this field static or non-static?

2) Attach the "Delete Points" button's clicked signal to a slot that changes the value of the PlotWindow's delete\_mode\_ field to the opposite of the value that it previously had. Use the ui->centralWidget->setCursor() method (<http://doc.qt.io/qt-5/qt.html#CursorShape-enum>) to change the appearance of the cursor to something else when the user is in delete mode. Make sure to change it back when the user is no longer in delete mode!