Name: Caleb Latimer

Student access ID: ej1297

Project: easyPaint

Date: 3/7/2017

Group Number: 1

**Change Request 2: Horizontal Line, 45-degree Line and Vertical Line**

1. **Change Request and concepts*:***

Add an additional capability to the line tool, which can be used to draw a strait horizontal or a vertical line or a 45-degree line (all kinds must be supported).

The users should press shift to draw these special lines.

Significant concepts are:

* Line tool – high possibility
* Shift key handler – possible, but not likely

1. **Concept Location:**

As per the requirements for change request 2 I made sole use of class dependency relations to isolate the concept location and perform impact analysis.

**Table 1. Dependency Search: lineTool**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class/file name** | **Tool used** | **Mark** | **Comments** |
| lineInstrument | Go to definition | Located | Contains click handlers for line tool, shift click handler will be added here, paint() will be what reflects the vertical,horizontal and 45 degree line measurements |
| Abstractinstrument | Go to definition | Unchanged | Base class used for deriving instruments, change does not affect |
| Imagearea | Find all references | unchanged | Calls an instance of lineInstrument but does not appear to change anything with it’s functionality |

****

1. **Impact Analysis:**

**Table 1. The list of all the classes visited during impact analysis.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Class name** | **Tool used** | **Mark** | **Comments** |
| 1 | AbstractInstrument | View call hierarchy | Unchanged | Interface, does not change |
| 2 | lineInstrument | View call hierarchy | Impacted | Changes will affect class entirely |
| 3 | Imagearea | View call hierarchy | Impacted | Changes will change rendering |
| 4 | QPainter | View call hierarchy | Unchanged | Does not tamper with how paint works |
| 5 | DataSingleton | View call hierarchy | Unchanged | Used for setting up pen type and color type |
| 6 | QMouseEvent | View call hierachy | Unchanged | unaffected |
| 7 | QImage | View call hierachy | Unchanged | Not affected |
| 8 | QFlags<enum> | View call hierachy | Unchanged | Not affected |
| 9 | QWidget | View call hierachy | Unchanged | Not affected |



1. **Prefactoring: No prefactoring was done in this request**
2. **Actualization:**

**Table 3. Actualization Summary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Code Files** | | | | | |
| Visited | Changed | Added | Propagating | Unchanged | Added to Changed Set |
| 2 | 1 | 0 | 2 | 2 | 0 |

**Table 4. Actualization Code Files**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Code File Name** | **Task** | **Lines of Code** | | |
| **Added** | **Deleted** | **Total** |
| 1 | lineInstrument | Implement shift key handler that rotated lines accordingly with the request | 18 | 0 | 18 |

1. **Postfactoring: No postfactoring was done in this request**
2. **Verification:**

Implemented unit test for lineinstrument functionality to do the following:

* Draw a normal line
* Rotate normal line
* Hold shift to transform line horizontal (left-right)
* Hold shift to transform line horizontal (right-left)
* Hold shift to transform line vertical (top-bottom)
* Hold shift to transform line vertical (bottom-top)
* Hold shift to transform line 45degree(topLeft-bottomRight)
* Hold shift to transform line 45degree (bottomLeft-topRight)
* Hold shift to transform line 45degree (topRight-bottomLeft)
* Hold shift to transform line 45degree (bottomRight-topLeft)

**Table 6. Statement Verification**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Code File Name** | **Coverage of Application** | | | **Tests Failed** | **Bugs Found** |
| **Total Statements** | **Covered Statements** | **%** |
| 1 | Lineinstrument.cpp | 10 | 10 | 60 | 4 | 4 |

1. **Sources:** I referenced the following documentation:

<http://stackoverflow.com/questions/7394660/check-if-a-key-is-down-with-qt>

<http://doc.qt.io/qt-4.8/qapplication.html#keyboardModifiers>

<http://doc.qt.io/qt-4.8/qapplication.html#queryKeyboardModifiers>

<http://www.cplusplus.com/reference/cmath/pow/>

<http://www.purplemath.com/modules/distform.htm>

<http://doc.qt.io/qt-5/qpainter.html#rotate>

<http://doc.qt.io/qt-5/qpainter.html>

<http://doc.qt.io/qt-5/qkeyevent.html>

<http://doc.qt.io/qt-5/qwidget.html#keyPressEvent>

1. **Highlighted Source Code:**

#include<QApplication>

void LineInstrument::paint(ImageArea &imageArea, bool isSecondaryColor, bool)

{

QPainter painter(imageArea.getImage());

if(isSecondaryColor)

{

painter.setPen(QPen(DataSingleton::Instance()->getSecondaryColor(),

DataSingleton::Instance()->getPenSize() \* imageArea.getZoomFactor(),

Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin));

}

else

{

painter.setPen(QPen(DataSingleton::Instance()->getPrimaryColor(),

DataSingleton::Instance()->getPenSize() \* imageArea.getZoomFactor(),

Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin));

}

if(mStartPoint != mEndPoint) // here is where the line is drawn

{

int deltaX, deltaY;

// my modifications start here

if (QGuiApplication::queryKeyboardModifiers().testFlag(Qt::ShiftModifier)) { // check if shift key is active

// Compute change in x and change in y with absolute value to prevent faulty logic

deltaX = abs(mEndPoint.x() - mStartPoint.x());

deltaY = abs(mEndPoint.y() - mStartPoint.y());

if (deltaX > deltaY){

// transform to a horizontal line

mEndPoint.setY(mStartPoint.y()); // rotate 180 making a horizontal line

}

else if (deltaX < deltaY){

// transform to a vertical line

mEndPoint.setX(mStartPoint.x()); // rotate 90 making a vertical line

}

else if(deltaX == deltaY){

// transform to a 45 degree line

double pi = acos(-1); // Initialize pi

double angle = 45 / 180.0 \* pi; // sets angle to 45 degrees but in radians

double LineLength = sqrt((pow(deltaX, 2) + pow(deltaY, 2))); // finds the distance of the line

mEndPoint.setX(cos(angle)\*LineLength + mStartPoint.x());

mEndPoint.setY(sin(angle)\*LineLength + mStartPoint.y());

}

else{

// Not a special line do nothing

}

painter.drawLine(mStartPoint, mEndPoint); // let the line be drawn

}// and end here

painter.drawLine(mStartPoint, mEndPoint); // draw normal line if shift is not pressed

}

if(mStartPoint == mEndPoint)

{

painter.drawPoint(mStartPoint);

}

imageArea.setEdited(true);

//int rad(DataSingleton::Instance()->getPenSize() + round(sqrt((mStartPoint.x() - mEndPoint.x()) \*

// (mStartPoint.x() - mEndPoint.x()) +

// (mStartPoint.y() - mEndPoint.y()) \*

// (mStartPoint.y() - mEndPoint.y()))));

//mPImageArea->update(QRect(mStartPoint, mEndPoint).normalized().adjusted(-rad, -rad, +rad, +rad));

painter.end();

imageArea.update();

}