Name: Caleb Latimer

Student access ID: ej1297

Project: easyPaint

Date: 2/19/2017

Group Number: 1

**Dash Line**

1. **Change Request and concepts**: In this change request I will be implementing the dashed line feature. It will add a new instrument to the easyPaint project allowing for dashed lines to be drawn when in use.
2. **Concept Location:**

Explain the methodology that you have used to **locate each significant concept (use either dependency search or grep search)** that was part of your change request.

**Table 1. Dependency Search: Dash Line**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class/file name** | **Tool used** | **Mark** | **Comments** |
| Main | Go to definition | Propogating | Doesn’t change itself but will call things that are going to be affected by the new function |
| Mainwindow | Go to definition | Changed | The class is greatly affected by the change due to aggregate classes. New instrument needs to be added to this class to the toolbar, initializeMenu, updateShortcuts, |
| Toolbar | Go to definition | Changed | The class is strongly affected because adding a new button for the instrument needs to be handled in here |

1. **Impact Analysis:**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Class name** | **Tool used** | **Mark** | **Comments** |
| 1 | Main | Go to definition | Propogating | Doesn’t change itself but will call things that are going to be affected by the new function |
| 2 | Mainwindow | Go to definition | Changed | The class is greatly affected by the change due to aggregate classes. New instrument needs to be added to this class to the toolbar, initializeMenu, updateShortcuts, |
| 3 | Toolbar | Go to definition | Changed | The class is strongly affected because adding a new button for the instrument needs to be handled in here |
| 4 | Shortcut edit | View class hierarchy | Unchanged | Has functions for left click and key press but these won’t be impacted by my changed |
| 5 | LineInstrument | View Class hierarchy | Unchanged | Has functions for painting in an image but nothing to do with lines or dashed line. Won’t be impacted by the change |



1. **Prefactoring:**

**Table 2. Prefactoring Code Files**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Code File Names** | **Refactoring Issue** | **Lines of Code** | | |
| **Added** | **Deleted** | **Total** |
|  | dashedLineInstrument | New component | 125 | 0 | 125 |

1. **Actualization:**

**Table 3. Actualization Summary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Code Files** | | | | | |
| Visited | Changed | Added | Propagating | Unchanged | Added to Changed Set |
| 15 | 11 | 2 | 2 | 1 | 2 |

**Table 4. Actualization Code Files**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Code File Name** | **Task** | **Lines of Code** | | |
| **Added** | **Deleted** | **Total** |
| 1 | mainwindow | Dashedline implementation | 7 | 0 | 7 |
| 2 | toolbar | Function for dashedline | 1 | 0 | 8 |
| 3 | Dashed.png | Icon image added for toolbar/menu | 0 | 0 | 8 |
| 4 | easyPaintnums | Enum added for dashed | 1 | 0 | 9 |
| 5 | Datasingleton | Shortcut added for dashed | 2 | 0 | 11 |
| 6 | imagearea | Cases added for dashed | 2 | 0 | 13 |
| 7 | dashedLineInstrument | New class added for implementing dashedline tool | 125 | 0 | 138 |

1. **Postfactoring:**

**Table 5. Postfactoring Code Files**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Code File Names** | **Refactoring Issue** | **Lines of Code** | | |
| **Added** | **Deleted** | **Total** |
| 1 | Toolbar | Functionality unneeded and interferes with layout of mainwindow | 0 | 1 | 137 |
| 2 | Dashed.png | Unnecessary file for change request | 0 | 0 | 137 |

1. **Verification:**

**Table 6. Statement Verification**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Code File Name** | **Coverage of Application** | | | **Tests Failed** | **Bugs Found** |
| **Total Statements** | **Covered Statements** | **%** |
| 1 | cR1 | 10 | 10 | 85 | 2 | 2 |

1. **Sources:** QObject bug resolution link - [stackoverflow](http://stackoverflow.com/questions/14170770/unresolved-external-symbol-public-virtual-struct-qmetaobject-const-thiscal%23)., Source for toolabar icon for dashed line image pulled with “creative commons license” [thenounproject](https://thenounproject.com/term/dotted-line/591150/) !NOTE: Image was not used and was removed in post-factoring
2. **Highlighted Source Code:**

**Mainwindow.cpp - initializeMenu()**

QAction \*mdashedLineAction = new QAction(tr("Dashed"), this);

mdashedLineAction->setCheckable(true);

mdashedLineAction->setIcon(QIcon(":/media/instruments-icons/dashed.png"));

connect(mdashedLineAction, SIGNAL(triggered(bool)), this, SLOT(instumentsAct(bool)));

mInstrumentsMenu->addAction(mdashedLineAction);

mInstrumentsActMap.insert(DASHED, mdashedLineAction);

**Mainwindow.cpp - updateShortcuts()**

mInstrumentsActMap[DASHED]->setShortcut(DataSingleton::Instance()->getInstrumentShortcutByKey("Dashed"));

**Toolbar.h – class definition**

QToolButton \*mCursorButton, \*mEraserButton, \*mPenButton, \*mLineButton,

\*mColorPickerButton, \*mMagnifierButton, \*mSprayButton, \*mFillButton,

\*mRectangleButton, \*mEllipseButton, \*mCurveButton, \*mTextButton, \*mDashedButton;

**Toolbar.cpp - initializeItems()**

mDashedButton = createToolButton(mActMap[DASHED]);

**easyPaintNums.h- class declaration**

typedef enum

{

NONE\_INSTRUMENT = 0,

CURSOR,

ERASER,

PEN,

LINE,

COLORPICKER,

MAGNIFIER,

SPRAY,

FILL,

RECTANGLE,

ELLIPSE,

CURVELINE,

TEXT,

DASHED,

// Don't use it. (Used to know count of current instrument)

INSTRUMENTS\_COUNT

} InstrumentsEnum;

**Datasingleton.cpp - readSetting()**

mInstrumentsShortcuts.insert("Dashed", settings.value("/Shortcuts/Instruments/Dashed", "").value<QKeySequence>());

**Datasingleton.cpp - writeSetting()**

settings.setValue("/Shortcuts/Instruments/Dashed", mInstrumentsShortcuts["Dashed"]);

**dashedlineInstrument.h**

#ifndef DASHEDLINEINSTRUMENT\_H

#define DASHEDLINEINSTRUMENT\_H

#include"abstractinstrument.h"

#include <QtCore/QObject>

class DashedLineInstrument : public AbstractInstrument{

Q\_OBJECT

public:

explicit DashedLineInstrument(QObject \*parent = 0);

void mousePressEvent(QMouseEvent \*event, ImageArea &imageArea);

void mouseMoveEvent(QMouseEvent \*event, ImageArea &imageArea);

void mouseReleaseEvent(QMouseEvent \*event, ImageArea &imageArea);

protected:

void paint(ImageArea &imageArea, bool isSecondaryColor = false, bool additionalFlag = false);

};

#endif

**dashedlineInstrument.h**

#include "dashedLineInstrument.h"

#include"../imagearea.h"

#include "../datasingleton.h"

#include <QPen>

#include <QPainter>

#include <QImage>

DashedLineInstrument::DashedLineInstrument(QObject \*parent) :

AbstractInstrument(parent)

{

}

void DashedLineInstrument::mousePressEvent(QMouseEvent \*event, ImageArea &imageArea)

{

if (event->button() == Qt::LeftButton || event->button() == Qt::RightButton)

{

mStartPoint = mEndPoint = event->pos();

imageArea.setIsPaint(true);

mImageCopy = \*imageArea.getImage();

makeUndoCommand(imageArea);

}

}

void DashedLineInstrument::mouseMoveEvent(QMouseEvent \*event, ImageArea &imageArea)

{

if (imageArea.isPaint())

{

mEndPoint = event->pos();

imageArea.setImage(mImageCopy);

if (event->buttons() & Qt::LeftButton)

{

paint(imageArea, false);

}

else if (event->buttons() & Qt::RightButton)

{

paint(imageArea, true);

}

}

}

void DashedLineInstrument::mouseReleaseEvent(QMouseEvent \*event, ImageArea &imageArea)

{

if (imageArea.isPaint())

{

imageArea.setImage(mImageCopy);

if (event->button() == Qt::LeftButton)

{

paint(imageArea, false);

}

else if (event->button() == Qt::RightButton)

{

paint(imageArea, true);

}

imageArea.setIsPaint(false);

}

}

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

{

QPainter painter(imageArea.getImage());

if (isSecondaryColor)

{

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

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

Qt::DashLine, Qt::RoundCap, Qt::RoundJoin));

}

else

{

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

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

Qt::DashLine, Qt::RoundCap, Qt::RoundJoin));

}

if (mStartPoint != mEndPoint)

{

painter.drawLine(mStartPoint, mEndPoint);

}

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();

}

**Imagearea.cpp- constructor**

mInstrumentsHandlers[DASHED] = new DashedLineInstrument(this);

**Imagearea.cpp - header**

#include "instruments\dashedLineInstrument.h"

**Imagearea.cpp - drawCursor()**

switch(DataSingleton::Instance()->getInstrument())

{

case NONE\_INSTRUMENT: case LINE: case COLORPICKER: case MAGNIFIER: case SPRAY:

case FILL: case RECTANGLE: case ELLIPSE: case CURSOR: case INSTRUMENTS\_COUNT:

case CURVELINE: case TEXT: case DASHED:

break;

case PEN: case ERASER:

mPixmap->fill(QColor(0, 0, 0, 0));

break;

}

switch(DataSingleton::Instance()->getInstrument())

{

case NONE\_INSTRUMENT: case LINE: case COLORPICKER: case MAGNIFIER: case SPRAY:

case FILL: case RECTANGLE: case ELLIPSE: case CURSOR: case INSTRUMENTS\_COUNT:

case CURVELINE: case TEXT: case DASHED:

break;

**cmake**

sources/instruments/dashedLineInstrument.h

sources/instruments/dashedLineInstrument.cpp

**pro**

instruments/dashedLineInstrument.cpp \

instruments/dashedLineInstrument.h \