ECE 462 C++ and Java

Lab Exercise 03
Graphical User Interface using C++ and Qt

Yung-Hsiang Lu
yunglu@purdue.edu
Install Qt in Your Computer
Trolltech offers solutions for

- Cross-Platform Developers
- Industrial Embedded Developers
- Consumer Electronics & Mobile Device Vendors
- Mobile Application Developers
- Software Development Managers
- Open Source Developers

Qt JAMBI GETS HIGH MARKS FOR JAVA DEVELOPMENT
"... DELIVERS EXCEPTIONAL EASE OF DEVELOPMENT" – InfoWorld

Qt
Qt is a cross-platform application framework. It includes:

Qtopia
Qtopia is an application platform and UI for Linux-based mobile, CE and embedded devices. Qtopia

GUI using C++ and Qt
I’m Developing Commercial Software
Trolltech offers free 30-day full-featured evaluation versions of Qt and Qtoba, as well as a range of add-ons, plug-ins and snapshots for developers creating commercial software.

I’m Developing Open Source Software
Trolltech provides the open source community with software under the GPL and other open source licenses for creating open source software.

Start on the Right Foot
Remember, you must purchase a Commercial License before beginning a proprietary development project. The Commercial License does not allow you to incorporate code developed with the...
Open Source Downloads

A Note on Licensing

Trolltech’s dual license policy provides versions of Qt and Qtopia under the terms of the GNU General Public License (GPL, version 2.0). Commercial, proprietary source projects working with Qt must always use the commercial version of Qt throughout their development process. The GPL versions can be downloaded free of charge and are provided without support or warranty. They are subject to the terms and conditions of the GPL. Read more in the Open Source download FAQ.

Start on the Right Foot
Remember, you must purchase a Commercial License before beginning a proprietary development project. The Commercial License does not allow you to incorporate code developed with the Open Source Editions of Trolltech software into a proprietary project. This means that you cannot begin proprietary development with the Open Source Edition, and then purchase a Commercial License later.

If you intend to use Qt or Qtopia for a proprietary project,
Install Qt in Linux
# Qt Open Source Edition for C++ Developers

<table>
<thead>
<tr>
<th>download</th>
<th>description</th>
</tr>
</thead>
</table>
| ![Download](qt_icon.png) | **Qt Open Source Edition for C++ Developers: Windows Download**
Qt is a cross-platform application development framework. The Open Source Edition of Qt is available to open source developers under the terms of the GPL version 2.0, and is not intended for commercial, proprietary source projects. |
| ![Download](qt_icon.png) | **Qt Open Source Edition for C++ Developers: Mac Download**
Qt is a cross-platform application development framework. The Open Source Edition of Qt is available to open source developers under the terms of the GPL version 2.0, and is not intended for commercial, proprietary source projects. |
| ![Download](qt_icon.png) | **Qt Open Source Edition for C++ Developers: X11 Download**
Qt is a cross-platform application development framework. The Open Source Edition of Qt is available to open source developers under the terms of the GPL version 2.0, and is not intended for commercial, proprietary source projects. |

## Provide Feedback on Pre-Release Software

<table>
<thead>
<tr>
<th>download</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="qt_icon.png" alt="Download" /></td>
<td><strong>Qt 4.4 Technology Preview</strong></td>
</tr>
</tbody>
</table>

---

**for Linux**
Install in Linux

Freie Universität Berlin, Berlin (Germany)
FTP:
- ftp://ftp.fu-berlin.de/unix/X11/gui/Qt/source/qt-x11-opensource-src-4.3.3.tar.gz

Download from ftp.trolltech.com:
- qtx11-opensource-src-4.3.3.tar.gz (approx. 42 Mb)

Packages
Most Linux and FreeBSD distributors offer adapted binary or source packages in the system's native format (like .rpm or .deb). These are probably easier to install and will fit well into your system.

Please check the download site of your vendor! Some operating system vendors and community sites that ship Qt Open Source Edition packages are:
- Conectiva
- Debian
- FreeBSD
- Mandriva
- Red Hat
- Slackware
You have chosen to open

qt-x11-opensource-src-4.3.3.tar.gz

which is a gzip archive

What should Firefox do with this file?

- Open with Archive Manager (default)
- Save to Disk

Do this automatically for files like this from now on.

[Cancel] [OK]
[Linux ~/Download/] tar xzf qt-x11-opensource-src-4.3.3.tar.gz
[Linux ~/Download/] cd qt-x11-opensource-src-4.3.3/
[Linux ~/Download/qt-x11-opensource-src-4.3.3/] ./configure --prefix=/home/yunl u/Programs/Qt433

This is the Qt/X11 Open Source Edition.

You are licensed to use this software under the terms of either
the Q Public License (QPL) or the GNU General Public License (GPL).

Type 'Q' to view the Q Public License.
Type 'G' to view the GNU General Public License.
Type 'yes' to accept this license offer.
Type 'no' to decline this license offer.

Do you accept the terms of either license? yes
Qt is now configured for building. Just run 'make'.
Once everything is built, you must run 'make install'.
Qt will be installed into /home/yunglu/Programs/Qt433

To reconfigure, run 'make confclean' and 'configure'.

[Linux ~/Download/qt-x11-opensource-src-4.3.3/]
GUI using C++ and Qt
GUI using C++ and Qt

execute Qt designer
To compile a Qt program:
1. qmake -project
2. qmake
3. make
GUI using C++ and Qt
Install Qt in Windows
Install in Windows

Welcome to the Qt OpenSource 4.3.3 Setup Wizard

This wizard will guide you through the installation of Qt OpenSource 4.3.3.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.
You are now installing the Open Source Edition of Qt. This Edition can only be used for development of Open Source software.

The Open Source Edition supports the MinGW compiler and is licensed under the GNU GPL license. Using the Open Source Edition will require you to distribute your Qt-based software under the GPL license.

The commercially licensed Editions of Qt support the Microsoft Visual Studio compilers and optionally the Visual Studio IDE. We offer free evaluations at:

http://www.trolltech.com/developer/downloads/qt

To read more about Trolltech’s licensing, please go to:

http://www.trolltech.com/company/model
Qt OpenSource 4.3.3 Setup

License Agreement
Please review the license terms before installing Qt OpenSource 4.3.3.

Press Page Down to see the rest of the agreement.

The Qt GUI Toolkit is Copyright (C) 1994-2007 Trolltech ASA.
You may use, distribute and copy the Qt GUI Toolkit under the terms of
GNU General Public License version 2, which is displayed below.

If you accept the terms of the agreement, select the first option below. You must accept the
agreement to install Qt OpenSource 4.3.3. Click Next to continue.

- I accept the terms in the License Agreement
- I do not accept the terms in the License Agreement

Nullsoft Install System v2.18

< Back   Next >   Cancel
Choose Components
Choose which features of Qt OpenSource 4.3.3 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- Qt 4.3.3
- File Associations

Space required: 233.1MB

Nullsoft Install System v2.18

< Back  Next >  Cancel
Choose Install Location
Choose the folder in which to install Qt OpenSource 4.3.3.

Setup will install Qt OpenSource 4.3.3 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder
C:\yunglu\Qt

Space required: 233.1MB
Space available: 8.1GB

Nullsoft Install System v2.18
Choose Start Menu Folder
Choose a Start Menu folder for the Qt OpenSource 4.3.3 shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Qt by Trolltech v4.3.3 (OpenSource)
Accessories
Administrative Tools
ArcSoft Collage Creator
Camtasia Studio 5
Canon Printer Uninstaller
Creative
Cygwin
DeductionPro 2006
Dell Accessories
Dell Picture Studio v2.0
Dell Support
Dragon NaturallySpeaking 9.5
GUI using C++ and Qt
MinGW License Agreement
Please review the license terms before installing MinGW.

MinGW License Information

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. GPL permits and encourages
these actions.

If you accept the terms of the agreement, click I Agree to continue. You must accept the
agreement to install Qt OpenSource 4.3.3.
MinGW Download

Downloading MinGW from server...
Installing
Please wait while Qt OpenSource 4.3.3 is being installed.

Extract: doc\html\demos-textedit.html ... 100%

Show details

Nullsoft Install System v2.18

< Back  Next >  Cancel
GUI using C++ and Qt
Installation Complete
Setup was completed successfully.

- Patching paths in qmake...
- Patching paths in core...
- Patching license information...
- Create shortcut: C:\Documents and Settings\All Users\Start Menu\Programs\Qt by Trace...
- Create shortcut: C:\Documents and Settings\All Users\Start Menu\Programs\Qt by Trace...
- Installing MinGW into C:\yunglu\Qt\MinGW
- Copying MinGW runtime...
- Created uninstaller: C:\yunglu\Qt\uninst.exe
- Create shortcut: C:\Documents and Settings\All Users\Start Menu\Programs\Qt by Trace...
- Create shortcut: C:\Documents and Settings\All Users\Start Menu\Programs\Qt by Trace...
- Output folder: C:\yunglu\Qt
- Completed
Completing the Qt OpenSource 4.3.3 Setup Wizard

Qt OpenSource 4.3.3 has been installed on your computer.

Click Finish to close this wizard.

- Run Examples and Demos
- Show Documentation

Finish Cancel
Qt Examples and Demos

Qt is supplied with a number of example applications and demonstrations that have been written to provide developers with examples of the Qt API in use, highlight good programming practice, and showcase features found in each of Qt's core technologies.

The example and demo launcher can be used to explore the different categories available. It provides an overview of each example, lets you view the documentation in Qt Assistant, and is able to launch examples and demos.

Documentation for examples can be found in the Tutorial and Examples section of the Qt documentation.
Widgets

Qt comes with a large range of standard widgets that users of modern applications have come to expect.

You can also develop your own custom widgets and controls, and use them alongside standard widgets.

It is even possible to provide custom styles and themes for widgets that can be used to change the appearance of standard widgets and appropriately written custom widgets.
Tetrix

The Tetrix example is a Qt version of the classic Tetrix game.
GUI using C++ and Qt
Qt Documentation
Online Reference Documentation

<table>
<thead>
<tr>
<th>Qt</th>
<th>Qtopía</th>
<th>QSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ Application Development Framework</td>
<td>Application Platform and User Interface for Embedded Linux</td>
<td>Qt Script for Applications</td>
</tr>
<tr>
<td>Qt 4.3 / Qtopía Core 4.3</td>
<td>Qtopía 4.3 Beta</td>
<td>QSA 1.2</td>
</tr>
<tr>
<td>Latest Qt 4.3 snapshot</td>
<td>Latest Qtopía 4.3 snapshot</td>
<td>QSA 1.1</td>
</tr>
<tr>
<td>Latest Qt unstable snapshot</td>
<td>Qtopía 4.2</td>
<td>QSA 1.0</td>
</tr>
<tr>
<td>Qt 4.2 / Qtopía Core 4.2</td>
<td>Qtopía 4.1</td>
<td></td>
</tr>
<tr>
<td>Qt 4.1 / Qtopía Core 4.1</td>
<td>Qtopía 2.2</td>
<td></td>
</tr>
<tr>
<td>Qt 4.0</td>
<td>Qtopía 2.1</td>
<td></td>
</tr>
<tr>
<td>Qt 3.3</td>
<td>Qtopía 2.0</td>
<td></td>
</tr>
<tr>
<td>Qt 3.2</td>
<td>Qtopía 1.7</td>
<td></td>
</tr>
<tr>
<td>Qt 3.1</td>
<td>Qtopía 1.6</td>
<td></td>
</tr>
<tr>
<td>Qt 3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Qt 2.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Qt Jambi</th>
<th>Qt Quarterly</th>
<th>Addons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qt for Java Development</td>
<td>C++ and Qt Developers Newsletter</td>
<td>Components and Tools for Qt</td>
</tr>
</tbody>
</table>
# Qt Reference Documentation (Open Source Edition)

**Note:** This edition is for the development of Free and Open Source software only; see Qt Commercial Editions.

## Getting Started
- What's New in Qt 4.3
- How to Learn Qt
- Installation
- Tutorial and Examples
- Porting from Qt 3 to Qt 4

## General
- About Qt
- About Trolltech
- Commercial Edition
- Open Source Edition
- Frequently Asked Questions

## Developer Resources
- Mailing Lists
- Qt Community Web Sites
- Qt Quarterly
- How to Report a Bug
- Other Online Resources

## API Reference
- All Classes
- Main Classes
- Grouped Classes
- Annotated Classes
- Qt Classes by Module
- Inheritance Hierarchy
- All Functions
- Qtopia Core
- All Overviews and HOWTOs
- Qt Widget Gallery

## Core Features
- Signals and Slots
- Object Model
- Layout Management
- Paint System
- Graphics View
- Accessibility
- Tool and Container Classes
- Internationalization
- Plugin System
- Interprocess Communication

## Key Technologies
- Multithreaded Programming
- Main Window Architecture
- Rich Text Processing
- Model/View Programming
- Style Sheets
- Network Module
- OpenGL Module
- SQL Module
- SVG Module
- XML Module
GUI using C++ and Qt
Qt 4.3.3

C:\yunglu\Qt\examples\mainwindows\application>cd release
C:\yunglu\Qt\examples\mainwindows\application\release>application.exe
C:\yunglu\Qt\examples\mainwindows\application\release>
Sample Qt Program
```c++
#include <QApplication>
#include "userinterface.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    UserInterface uiWin(argc, argv);
    uiWin.show();
    return app.exec();
}
```

```c++
// userinterface.h
#ifndef USERINTERFACE_H
#define USERINTERFACE_H
#include <QtGui>
class UserInterface : public QMainWindow
{
public:
    UserInterface(int argc, char * argv[]);
    ~UserInterface();
};
#endif
```
C:\yunglu\teaching\ece462\lab03\dir
Volume in drive C has no label.
Volume Serial Number is 3014-7DB5

Directory of C:\yunglu\teaching\ece462\lab03

01/28/2008  01:00  PM          <DIR>          .
01/28/2008  01:08  PM          <DIR>          ..
01/28/2008  10:03  AM          130  userinterface.cpp
01/28/2008  10:09  AM          220  userinterface.h
3 File(s)                       564 bytes
2 Dir(s)   15,746,973,696 bytes free
GUI using C++ and Qt
C:\yunglu\teaching\ece462\lab03>qmake
C:\yunglu\teaching\ece462\lab03>dir
  Volume in drive C has no label.
  Volume Serial Number is 3014-7DB5

Directory of C:\yunglu\teaching\ece462\lab03
01/28/2008  01:10 PM    <DIR>  .
01/28/2008  01:10 PM    <DIR>  ..
01/28/2008  01:10 PM    <DIR>  debug
01/28/2008  01:09 PM    355 lab03.pro
01/28/2008  10:02 AM    214 main.cpp
01/28/2008  01:10 PM    5,676 Makefile
01/28/2008  01:10 PM    5,064 Makefile.Debug
01/28/2008  01:10 PM    5,115 Makefile.Release
01/28/2008  01:10 PM    <DIR>  release
01/28/2008  10:03 AM    130 userinterface.cpp
01/28/2008  10:09 AM    220 userinterface.h
7 File(s)       16,774 bytes
4 Dir(s)   15,746,949,120 bytes free

C:\yunglu\teaching\ece462\lab03>
1. qmake –project
2. qmake
3. make
C:\yunglu\teaching\ece462\lab03\debug>lab03.exe
C:\yunglu\teaching\ece462\lab03\debug>
Develop Qt Project in Netbeans
Creates a new project that uses an existing makefile or configure script and existing sources. It uses your makefile to build the project.
Choose "Using an existing makefile" if you use an already existing makefile to build your code. Choose "Using a makefile generated by a configure script" if you run a configure script to generate the makefile you use to build your code.
GUI using C++ and Qt

Working Directory is the directory in which the Build and Clean commands are executed. The Build and Clean commands are commands that map the IDE’s Build and Clean actions to shell commands. They could be executing an existing Makefile like "make -f Makefile" and "make -f Makefile clean". Build Result is the full path to the output of the Build Command and could be a path to an executable or library.
GUI using C++ and Qt
GUI using C++ and Qt
GUI using C++ and Qt
Add Buttons and Labels
/ userinterface.h
#define USERINTERFACE_H
#include <QtGui>
class PlayField;
class UserInterface : public QMainWindow
{
    Q_OBJECT
public:
    UserInterface(int argc, char * argv[]);
~UserInterface();
private slots:
    void startGame();
    void exitGame();
private:
    void createControl();
    void createStatus();

    QPushButton * ui_startButton;
    QWidget * ui_centralWidget;
    QGroupBox * ui_controlGroup;
    QGroupBox * ui_statusGroup;
    QLabel * ui_positionLabel;
    QLabel * ui_directionLabel;
    QLabel * ui_leftScoreLabel;
    QLabel * ui_rightScoreLabel;
    QLabel * ui_ballLeftLabel;
    PlayField * ui_playField;
};
#endif
```cpp
#include "userinterface.h"
#include "playfield.h"
#include <iostream>
using namespace std;

UserInterface::UserInterface(int argc, char * argv[]) {
    createControl();
    createStatus();
    ui_centralWidget = new QWidget;
    QVBoxLayout * mainLayout = new QVBoxLayout;
    ui_playField = new PlayField;
    ui_centralWidget -> setLayout(mainLayout);
    mainLayout ->addWidget(ui_playField);
    mainLayout ->addWidget(ui_controlGroup);
    mainLayout ->addWidget(ui_statusGroup);
    setCentralWidget(ui_centralWidget);
    setWindowTitle(tr("Two-Player Breakout Game"));
}

UserInterface::~UserInterface() {
}

void UserInterface::createControl() {
    QVBoxLayout * controlLayout = new QVBoxLayout;
    QWidget *startExitWidget = new QWidget;
    QVBoxLayout * startExitLayout = new QVBoxLayout;
    ui_startButton = new QPushButton("Start Game");
    QPushButton * exitButton = new QPushButton("Exit");
    startExitLayout ->addWidget(ui_startButton);
    startExitLayout ->addWidget(exitButton);
```
```cpp
void UserInterface::createStatus()
{
    QGridLayout * statusLayout = new QGridLayout;
    QLabel * position = new QLabel("position: ");
    ui_positionLabel = new QLabel(" ");
    QLabel * direction = new QLabel("direction: ");
    ui_directionLabel = new QLabel(" ");
    QLabel * leftScore = new QLabel("score: ");
    ui_leftScoreLabel = new QLabel("0");
    QLabel * ballLeft = new QLabel("ball left: ");
    ui_ballLeftLabel = new QLabel("4");
    QLabel * rightScore = new QLabel("score: ");
    ui_rightScoreLabel = new QLabel("0");
    statusLayout -> addWidget(position, 0, 0);
    statusLayout -> addWidget(ui_positionLabel, 0, 1);
    statusLayout -> addWidget(direction, 0, 2);
    statusLayout -> addWidget(ui_directionLabel, 0, 3);
    statusLayout -> addWidget(ballLeft, 0, 4);
    statusLayout -> addWidget(ui_ballLeftLabel, 0, 5);
    statusLayout -> addWidget(leftScore, 1, 0);
    statusLayout -> addWidget(ui_leftScoreLabel, 1, 1);
}
```
statusLayout -> addWidget(leftScore, 1, 0);
statusLayout -> addWidget(ui_leftScoreLabel, 1, 1);
statusLayout -> addWidget(rightScore, 1, 4);
statusLayout -> addWidget(ui_rightScoreLabel, 1, 5);
ui_statusGroup = new QGroupBox(tr("Status"));
ui_statusGroup -> setLayout(statusLayout);

void UserInterface::startGame()
{
}

void UserInterface::exitGame()
{
  close();
}
GUI using C++ and Qt
Handle User Inputs (Signals)
Signals and Slots

Signals and slots are used for communication between objects. The signals and slots mechanism is a central feature of Qt and probably the part that differs most from the features provided by other frameworks.

- Introduction
- Signals and Slots
- A Small Example
- Building the Example
- Signals
- Slots
- Meta-Object Information
- A Real Example

signal = event (in Java)
slot = handler
Qt 4.3: Signals and Slots - Mozilla Firefox

```
connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)
connect(Object1, signal2, Object4, slot1)
```
Add Paddles and a Bouncing Ball
GUI using C++ and Qt
#ifndef _TWODVECTOR_H_
#define _TWODVECTOR_H_

// This class handles two-dimensional analytic geometry. C++ or
// Qt's vector classes are container classes.

class TwoDVector
{
public:
    TwoDVector(double x = 1.0, double y = 1.0, bool unit = true);
    void reflect(const TwoDVector & vin, TwoDVector & vout) const;
    // the object is the unit vector, given the incident vector, find
    // the reflective vector. using passing-by-reference to reduce the
    // overhead of copying objects
    double innerProduct(const TwoDVector & v2) const;
    double getX() { return tdv_x; }
    double getY() { return tdv_y; }
    void addVector(const TwoDVector & vec);
    void print();

private:
    void makeUnit(); // keep it a unit vector
    double tdv_x;
    double tdv_y;
    bool unitVector;
};

extern void testVectorReflect();
// generate a set of vin and normal to test vout

#endif
```cpp
#include <iostream>
#include <math.h>
#include "twodvector.h"
using namespace std;

TwoDVector::TwoDVector(double x, double y, bool unit)
{
    tdv_x = x;
    tdv_y = y;
    unitVector = unit;
    if (unitVector == true) { makeUnit(); }
}

void TwoDVector::makeUnit()
{
    if ((tdv_x == 0) && (tdv_y == 0))
    {
        cerr << "ERROR, a (0, 0) vector" << endl; return;
    }
    if (tdv_x == 0)
    {
        if (tdv_y > 0) { tdv_y = 1.0; return; }
        tdv_y = -1.0;
        return;
    }
    if (tdv_y == 0)
    {
        if (tdv_x > 0) { tdv_x = 1.0; return; }
        tdv_x = -1.0;
        return;
    }
    double magnitude = sqrt(tdv_x * tdv_x + tdv_y * tdv_y);
    tdv_x /= magnitude;
    tdv_y /= magnitude;
--
```
```cpp
tdv_x /= magnitude;
    tdv_y /= magnitude;
}

double TwoDVector::innerProduct(const TwoDVector & v2) const
{
    return (tdv_x * v2.tdv_x + tdv_y * v2.tdv_y);
}

void TwoDVector::reflect(const TwoDVector & vin, TwoDVector & vout) const
{
    if ((unitVector == false) ||
        (vin.unitVector == false) ||
        (vout.unitVector == false))
    {
        cout << "ERROR! vector must unit" << endl; }

double s = -2 * innerProduct(vin);
    vout.tdv_x = vin.tdv_x + s * tdv_x;
    vout.tdv_y = vin.tdv_y + s * tdv_y;
    vout.makeUnit();
}

void TwoDVector::print()
{
    cout << "(" << tdv_x << "," << tdv_y << ")" << endl;
}

void TwoDVector::addVector(const TwoDVector & vec)
{
    tdv_x += vec.tdv_x;
    tdv_y += vec.tdv_y;
    if (unitVector == true) { makeUnit(); }
```

void testVectorReflect()
{
    cout << "test 1" << endl;
    TwoDVector vin1(-1, -1);
    TwoDVector nor1(0, 1);
    TwoDVector vout1(0, 1);
    nor1.reflect(vin1, vout1);
    vin1.print();
    nor1.print();
    vout1.print();

    cout << endl << "test 2" << endl;
    TwoDVector vin2(-1, 0);
    TwoDVector nor2(1, 1);
    TwoDVector vout2(0, 1);
    nor2.reflect(vin2, vout2);
    vin2.print();
    nor2.print();
    vout2.print();

    cout << endl << "test 3" << endl;
    TwoDVector vin3(-1, -sqrt(3));
    TwoDVector nor3(0, 1);
    TwoDVector vout3(0, 1);
    nor3.reflect(vin3, vout3);
    vin3.print();
    nor3.print();
}
/playfield.h
 ifndef _PLAYFIELD_H
 define _PLAYFIELD_H
 include <QtGui>
 include "twodvector.h"
 class PlayField: public QWidget
 {
 public:
 PlayField(QWidget *parent = 0);
 // no destructor since no attributed are created using new
 protected:
 void paintEvent(QPaintEvent *event);
 void keyPressEvent(QKeyEvent * event);
 void mouseMoveEvent(QMouseEvent * event);
 private:
 int pf_leftPaddleCenterY;
 int pf_rightPaddleCenterY;
 TwoDVector pf_ballPosition; // convert to int only for drawing
 TwoDVector pf_ballVelocity; // unit vector
 void updateBallPosition(int & bx, int & by);

 #endif

 --- playfield.h (C Abbrev)---L5--All---
/* playfield.cpp */
#include <QGui>
#include <iostream>
#include "playfield.h"
#include "constant.h"
using namespace std;
PlayField::PlayField(QWidget *parent): QWidget(parent)
{
    setMinimumSize(GC_fieldWidth, GC_fieldHeight);
    setMaximumSize(GC_fieldWidth, GC_fieldHeight);
    setMouseTracking(true);
    pf_leftPaddleCenterY = GC_paddleHalfHeight;
    pf_rightPaddleCenterY = GC_fieldHeight - GC_paddleHalfHeight;
    pf_ballPosition = TwoDVector(GC_fieldWidth/2, GC_ballRadius * 3, false);
}

void PlayField::paintEvent(QPaintEvent *event)
{
    int ww = width(); // widget's width
    int wh = height(); // widget's height
    // draw background
    QPainter painter(this);
    painter.setBrush(GC_backgroundColor);
    painter.drawRect(0, 0, ww, wh);
    // ****************
    // * draw paddles
    painter.setBrush(GC_leftPaddleColor);
    painter.drawChord(GC_leftPaddleCenterX,
                       pf_leftPaddleCenterY -
                        GC_paddleRadius,
                        GC_paddlediameter,
                        --\ playfield.cpp ~ (C++ Abbrev)--Ll--Top------------------
GUI using C++ and Qt

```cpp
GC_paddleRadius,
GC_paddleDiameter,
GC_paddleDiameter,
GC_leftPaddleStartAngle,
GC_paddleSpanAngle);
painter.setBrush(GC_rightPaddleColor);
painter.drawChord(GC_rightPaddleCenterX,
pf_rightPaddleCenterY -
GC_paddleRadius,
GC_paddleDiameter,
GC_paddleDiameter,
GC_rightPaddleStartAngle,
GC_paddleSpanAngle);

// ***********
// * draw ball
int bx, by;
updateBallPosition(bx, by);
painter.setBrush(GC_ballColor);
painter.drawEllipse(bx - GC_ballRadius,
by - GC_ballRadius,
GC_ballDiameter,
GC_ballDiameter);

void PlayField::keyPressEvent(QKeyEvent * event)
{
    switch (event -> key())
    {
    case Qt::Key_Up:
        break;
    case Qt::Key_Down:
```
```cpp
int newX = event -> x();
int newY = event -> y();
if (mY > newY) // mouse moved up
{
}
if (mY < newY)
   // cannot use "else" because this function is also called
   // when the mouse moves horizontally
{
}
mX = newX;
mY = newY;
// update();

void PlayField::updateBallPosition(int & bx, int & by)
{
   bx = (int) pf_ballPosition.getX();
   by = (int) pf_ballPosition.getY();
}
```
Move Ball
Track Mouse
Detect Collision
Law of Reflection

- S: surface of collision
- N: normal vector, $N \perp S$
- I: incident vector
- R: reflection vector
- a: angle between I and N
- b: angle between R and N
- law of reflection: $a = b$
- If N, I, and R are unit vectors ($|N| = |I| = |R| = 1$),
  $R - I = 2 \, N \cdot (-I)$, here $\cdot$ is the inner product.
  $\Rightarrow R = I - 2 \, N \cdot I$
Ball Hits Paddle

• Both are circles. The normal vector is the vector connecting the center of the ball and the center of the paddle.
• Collision occurs when the distance between the two centers is smaller than the sum of the radii.
void PlayField::mouseMoveEvent(QMouseEvent * event)
{
    // ignore X since the paddle moves only up and down
    static int lastY = event -> y();
    int newY = event -> y();
    while ((lastY > newY) && (pf_leftPaddleCenterY > GC_paddleHalfHeight))
    {
        // move paddle up but keep the paddle within the playfield
        pf_leftPaddleCenterY --;
        lastY --;
        update(); // redraw the playfield
    }
    while ((lastY < newY) &&
            (pf_leftPaddleCenterY < (GC_fieldHeight - GC_paddleHalfHeight)))
    {
        pf_leftPaddleCenterY ++;
        lastY ++;
        update();
    }
    lastY = newY; // this is necessary because the while block may break
    // before the paddle moves out of the playfield
}
void PlayField::updateBallPosition(int & bx, int & by)
{
// before the paddle moves out of the playfield

void PlayField::updateBallPosition(int & bx, int & by)
{
    bx = (int) pf_ballPosition.getX();
    by = (int) pf_ballPosition.getY();
    pf_ballPosition.addVector(pf_ballVelocity);
    double newBX = pf_ballPosition.getX();
    double newBY = pf_ballPosition.getY();
    // collision with paddles
    double diffX = newBX + GC_paddleHalfHeight;
    // already shifted right by paddleRadius from the left wall
    double diffY = newBY - pf_leftPaddleCenterY;
    TwoDVector normalVector;
    if ((diffX * diffX + diffY * diffY) <=
        (GC_ballPaddleRadiusSumSquare + 1))
        // + 1 to accommodate float-point imprecision
        
        // collide with left paddle
        normalVector = TwoDVector(diffX, diffY, true);
        findNewVelocity(normalVector);
        return; // ball cannot collide with a paddle and a wall simultaneously
    }

    // collision with walls
    if (newBX <= GC_ballRadius) // left wall
    {
        normalVector = TwoDVector(1, 0, true);
        --(Unix)-- playfield.cpp  (C++)--193--56%--
    }
if (newBY <= GC_ballRadius) // top wall
    normalVector = TwoDVector(0, 1, true);
    findNewVelocity(normalVector);
}
if (newBY >= (GC_fieldHeight - GC_ballRadius)) // bottom wall
    normalVector = TwoDVector(0, -1, true);
    findNewVelocity(normalVector);
}

void PlayField::findNewVelocity(const TwoDVector & normalVector)
{
    TwoDVector newVelocity;
    normalVector.reflect(pf_ballVelocity, newVelocity);
    double vx = newVelocity.getX();
    if ((vx > -0.01) && (vx < 0.01))
    { // almost moving vertically, add some horizontal velocity
        TwoDVector horizontal(vx * 10, 0, false);
        newVelocity.addVector(horizontal);
    }
    pf_ballVelocity = newVelocity;
}
GUI using C++ and Qt

```cpp
#include "userinterface.h"
#include "playfield.h"
#include "constant.h"
#include <iostream>

using namespace std;

UserInterface::UserInterface(int argc, char * argv[])
{
    createControl();
    createStatus();
    ui_centralWidget = new QWidget;
    QBoxLayout * mainLayout = new QVBoxLayout;
    ui_playField = new PlayField;
    ui_centralWidget->setLayout(mainLayout);
    mainLayout->addWidget(ui_playField);
    mainLayout->addWidget(ui_controlGroupBox);
    mainLayout->addWidget(ui_statusGroupBox);
    setCentralWidget(ui_centralWidget);
    setWindowTitle(tr("Two-Player Breakout Game"));
    ui_updateTimer = new QTimer;
    ui_updateTimer->setInterval(GC_updateDelay);
    ui_updateTimer->setSingleShot(false); // repeat the timer
    connect(ui_updateTimer, SIGNAL(timeout()), this, SLOT(updatePlayField()));
}

UserInterface::~UserInterface()
{
}

void UserInterface::createControl()
```
GUI using C++ and Qt

statusLayout -> addWidget(ui_leftScoreLabel, 1, 1);
statusLayout -> addWidget(rightScore, 1, 4);
statusLayout -> addWidget(ui_rightScoreLabel, 1, 5);
ui_statusGroup = new QGroupBox(tr("Status"));
ui_statusGroup -> setLayout(statusLayout);

void UserInterface::startGame()
{
    ui_updateTimer -> start();
}

void UserInterface::exitGame()
{
    // optional: ask the user whether the game should be terminated,
    // allow the user to cancel
    close();
}

void UserInterface::updatePlayField()
{
    ui_playField -> update();
}
Exercise Requirements

1. Make the right paddle follow the ball's vertical coordinate.
2. Detect collision between the right paddle and the ball and make the ball bounce.
3. Update the ball's position and direction.
Submission: A zip file of the CVS repository

Remember to commit all changes first.
Submit this exercise only.
Do not submit any other exercise.
Do not submit a wrong zip file.