

Практикум на ЭВМ

Qt. Отрисовка. Клавиатура. Ресурсы. Таймер.

Баев А.Ж.

Казахстанский филиал МГУ

04 декабря 2018

- 1) арканойд.
- 2) бильярд.

Отображать игровую сцену будем установкой изображения в QLabel методом

```
void QLabel::setPixmap(const QPixmap &)
```

The QPixmap class is an off-screen image representation that can be used as a paint device.

1. Создавать QPixmap будем динамически в конструкторе MainWindow.

```
class MainWindow : public QMainWindow {  
    QLabel *label;  
    QPixmap *pixmap;  
    Game game;  
public:  
    MainWindow(QWidget *parent = NULL);  
    ~MainWindow();  
};
```

```
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) {  
    label = new QLabel(this);  
    pixmap = new QPixmap(game.size);  
}  
MainWindow::~~MainWindow() {  
    delete label;  
    delete pixmap;  
}
```

2. Рисовать с помощью класса QPainter будем в методе MainWindow::draw(). После отрисовки на QPixmap отправляем QPixmap в QLabel.

```
class MainWindow : public QMainWindow {  
    ...  
public:  
    void draw();  
};
```

```
void MainWindow::frame () {  
    game.step();  
    game.draw(pixmap);  
    label->setPixmap(*pixmap);  
}
```

Игровую логику вынесем в отдельный класс Game. Для того, чтобы добавить класс, создаем game.cpp, game.h и добавляем соответствующие записи в .pro файл:

```
SOURCES += main.cpp mainwindow.cpp game.cpp
HEADERS += mainwindow.h game.h
QT += widgets
```

Не забываем сделать qmake.

В простой версии игры, понадобится сделать движение шарика с корректными отражениями

```
class Game {
    QSize size;
    QVector2D position;
    QVector2D velocity;
    QVector2D gravity;
    double tau;
    double radius;
    QPixmap *texture_ball;
    friend class MainWindow;

public:
    Game();
    ~Game();
    void step();
    void draw(QPixmap *pixmap);
};
```

```
Game::Game() {  
    size = QSize(600, 600);  
    position = QVector2D(size.width() / 2, size.height() / 2);  
    velocity = QVector2D(10.0, 20.0);  
    gravity = QVector2D(0.0, 10.0);  
    tau = 0.1;  
    radius = 50.0;  
    texture_ball = new QPixmap("pictures/ball.png");  
}
```

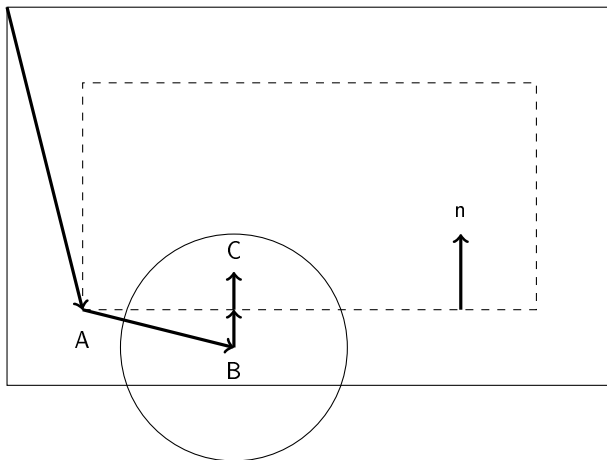

$$\frac{d\vec{x}}{dt} = \vec{v}$$

$$\frac{d\vec{v}}{dt} = \vec{a}$$

$$\hat{position} = position + \tau * velocity$$

$$\hat{velocity} = velocity + \tau * acceleration$$

Отражение точки B относительно прямой, проходящей через точку A и нормалью \vec{n} .



$$\vec{AB}_n = (\vec{AB}; \vec{n}) \vec{n}$$

$$\vec{AC} = \vec{AB} - 2 * \vec{AB}_n$$

```
void Game::step() {
    position = position + tau * velocity;
    velocity = velocity + tau * gravity;
    QVector2D normal[] = {
        QVector2D(1, 0),
        QVector2D(0, -1),
        QVector2D(-1, 0),
        QVector2D(0, 1) };
    QVector2D border[] = {
        QVector2D(radius, radius),
        QVector2D(radius, size.height() - radius),
        QVector2D(size.width() - radius, size.height() - radius),
        QVector2D(size.width() - radius, radius) };
```

```
qreal normal_factor;  
for (int i = 0; i < 4; i++) {  
    normal_factor = QVector2D::dotProduct(  
        position - border[i], normal[i]);  
    if (normal_factor < 0) {  
        position = position - 2 * normal_factor * normal[i];  
  
        normal_factor = QVector2D::dotProduct(  
            velocity, normal[i])  
        velocity = velocity - 2 * normal_factor * normal[i];  
    }  
}
```

Слоты

```
void start(int msec)
void start()
void stop()
```

Сигнал

```
void timeout()
```

Слоты

```
int QObject::startTimer(int interval)
void QObject::killTimer(int id)
```

Обработчик (обертка над сигналом)

```
void MyObject::timerEvent(QTimerEvent *event) {
    int id = event->timerId();
}
```

```
void QWidget::mousePressEvent(QMouseEvent *event) {
    QPoint lastPoint = event->pos();
}

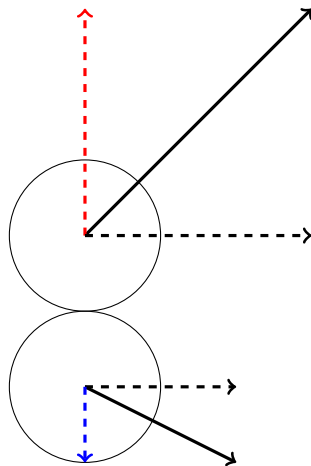
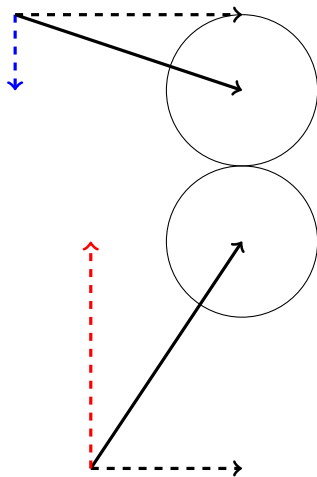
void QWidget::keyPressEvent(QKeyEvent *event) {
    QKey key = event->key();
    if (key == Qt::Key_Space) {
        ...
    }
}
```

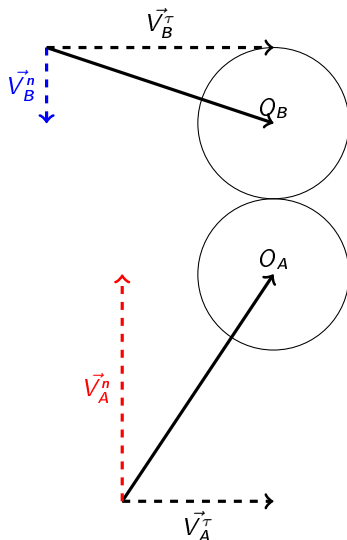
<http://doc.qt.io/qt-5/qtwidgets-widgets-scribble-example.html>

Упругий удар двух шаров

$$\vec{v}_1 + \vec{v}_2 = \vec{v}_1' + \vec{v}_2'$$

$$v_1^2 + v_2^2 = v_1'^2 + v_2'^2$$





$$\vec{e} = O_A \vec{O}_B / |O_A O_B|$$

$$\vec{V}_A^n = (\vec{V}_A; \vec{e}) \vec{e}$$

$$\vec{V}_A^\tau = \vec{V}_A - \vec{V}_A^n$$

$$\vec{V}_B^n = (\vec{V}_B; \vec{e}) \vec{e}$$

$$\vec{V}_B^\tau = \vec{V}_B - \vec{V}_B^n$$