

Человеко-машинное взаимодействие

Лекция 9

Мерзлякова Екатерина Юрьевна

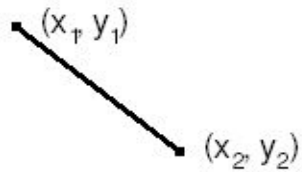
к.т.н. доцент ПМиК

2D графика. QPainter

перо (pen), кисть (brush) шрифт (font).

- `setPen()`
- `setBrush()`
- `setFont()`

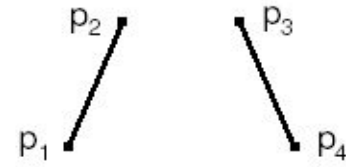
Методы класса QPainter, для рисования геометрических фигур



`drawLine()`



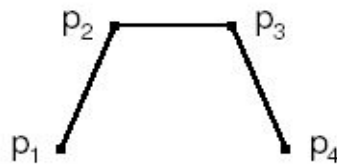
`drawPoints()`



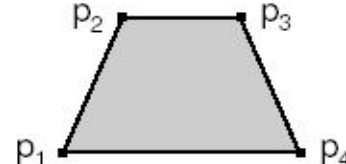
`drawLineSegments()`



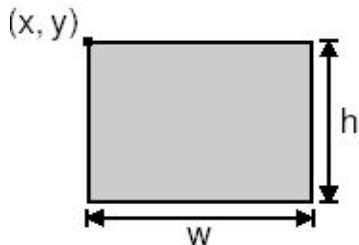
`drawCubicBezier()`



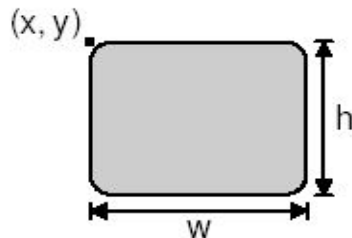
`drawPolyline()`



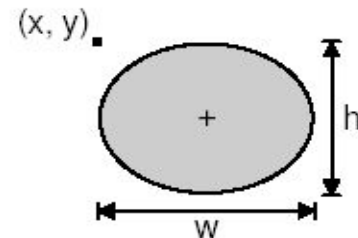
`drawPolygon()`



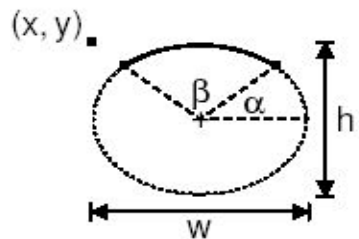
`drawRect()`



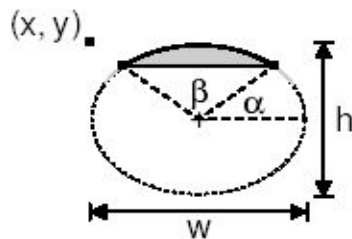
`drawRoundRect()`



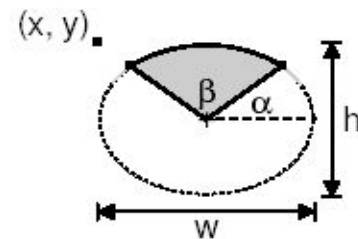
`drawEllipse()`



`drawArc()`



`drawChord()`

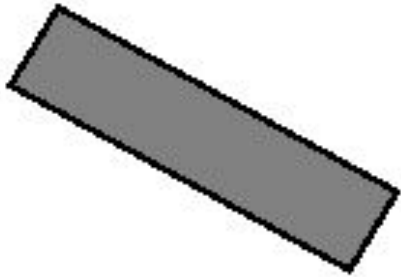


`drawPie()`

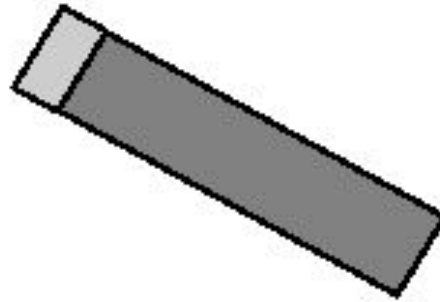
Стили пера.

	Толщина линий			
	1	2	3	4
NoPen				
SolidLine				
DashLine				
DotLine				
DashDotLine				
DashDotDotLine				

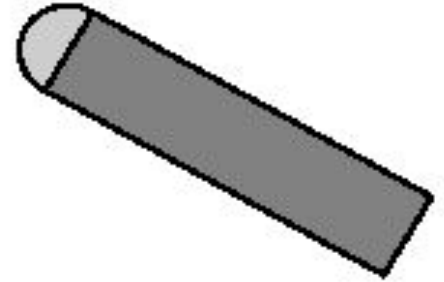
Стили оформления концов линий и углов.



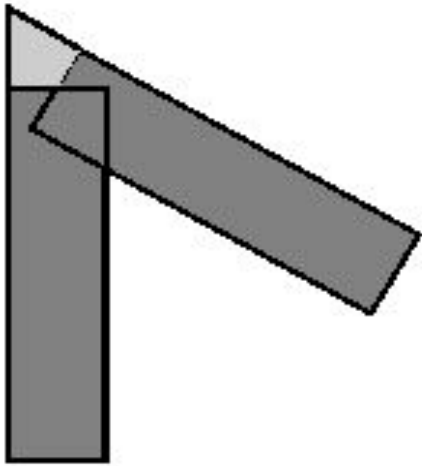
FlatCap



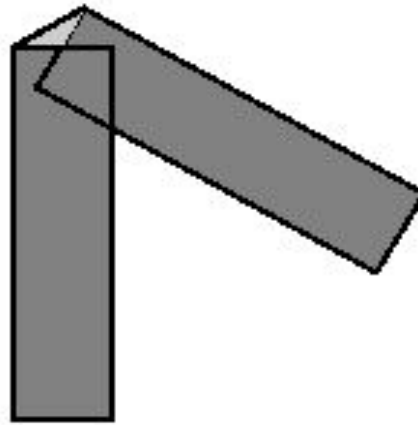
SquareCap



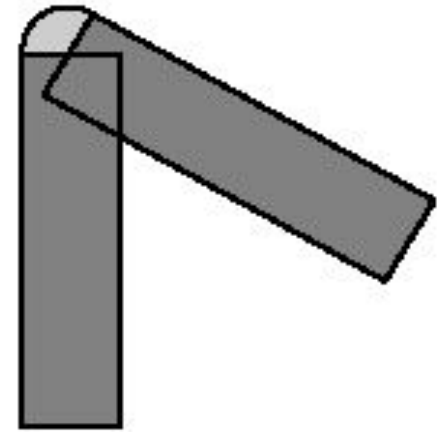
RoundCap



MiterJoin



BevelJoin



RoundJoin

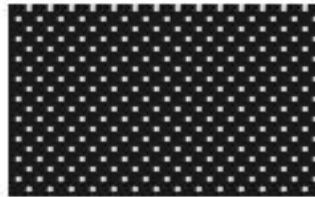
Стили кисти.



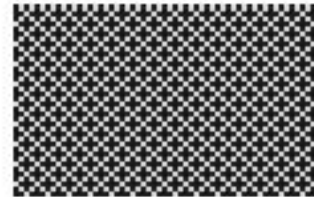
SolidPattern



Dense1Pattern



Dense2Pattern



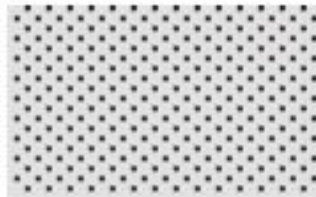
Dense3Pattern



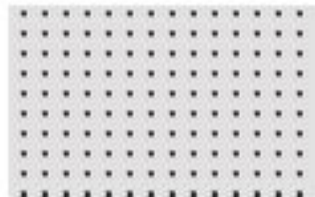
Dense4Pattern



Dense5Pattern



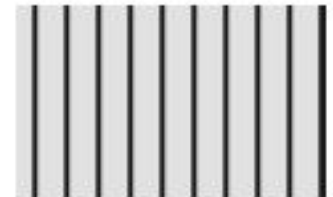
Dense6Pattern



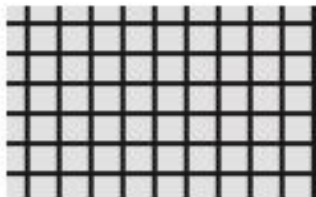
Dense7Pattern



HorPattern



VerPattern



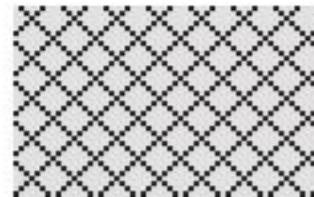
CrossPattern



BDiagPattern



FDiagPattern



DiagCross-



NoBrush

```
1 #include <QtGui/QApplication>
2 #include "mainwindow.h"
3 #include <QPainter>
4 #include <QLabel>
```

```
6 class MyWindow : public QMainWindow {
7     protected:
```

```
8     virtual void paintEvent(QPaintEvent* e) {
9         QMainWindow::paintEvent(e);
```

```
11         QPainter p(this);
12         p.setPen(QPen(Qt::red, 2, Qt::DotLine));
13         p.drawLine(0, 0, 100, 100);
```

```
14     }
```

```
15 };
```

```
16 int main(int argc, char *argv[])
```

```
17 {
18     QApplication a(argc, argv);
19     MyWindow w;
20     w.show();
21     return a.exec();
```

```
22 }
```



```
QPainter p(this);  
p.setPen(QPen(Qt::black, 3, Qt::DashDotLine));  
p.setBrush(QBrush(Qt::green, Qt::SolidPattern));  
p.drawEllipse(20, 20, 100, 60);
```




```
1 #include <QtGui/QApplication>
2 #include "mainwindow.h"
3 #include <QPainter>
4 #include <QPushButton>
5 #include <QMatrix>
6
7 class MyButton : public QPushButton {
8     protected:
9     virtual void paintEvent(QPaintEvent* e) {
10         QPushButton::paintEvent(e);
11         QPainter p(this);
12         p.setPen(QPen(Qt::black, 1, Qt::SolidLine));
13         p.setBrush(QBrush(Qt::green, Qt::SolidPattern));
14         p.drawEllipse(10, 7, 10, 6);
15     }
16 };
17
18 int main(int argc, char *argv[])
19 {
20     QApplication a(argc, argv);
21     MyButton b;
22     b.show();
23     return a.exec();
24 }
```



Параметры системы координат

- **область просмотра (viewport)**
- **окно (window)**
- **матрица преобразования (world matrix)**

- **матрица преобразования** позволяет выполнять изменение масштаба, вращение и сдвиг рисуемых элементов. Например, если необходимо нарисовать текст под углом 45 градусов, то можно написать следующий код:

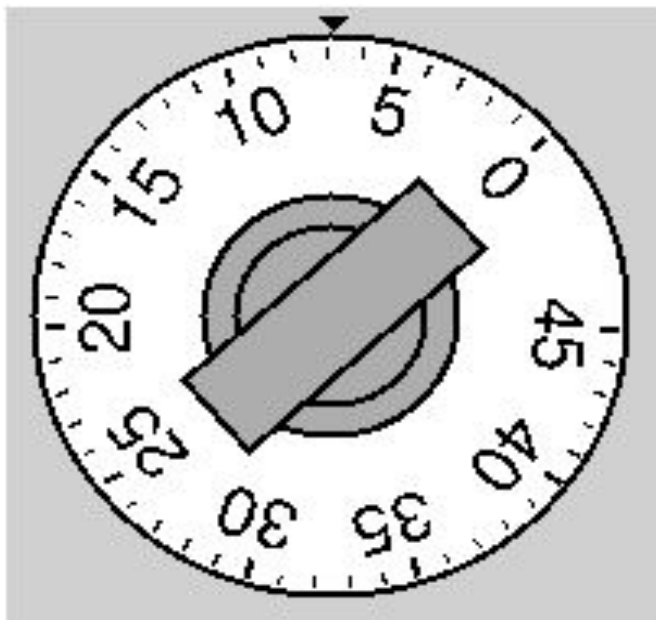
```
QMatrix matrix;  
matrix.rotate(45.0);  
matrix.translate(-40.0, -40.0);  
p.setWorldMatrix(matrix);  
p.drawText(rect(), Qt::AlignCenter, tr("Revenue"));
```



При необходимости, матрицу преобразований можно сохранить вызовом `saveWorldMatrix()` и затем восстановить вызовом `restoreWorldMatrix()`.

Реализация Таймера электропечи:

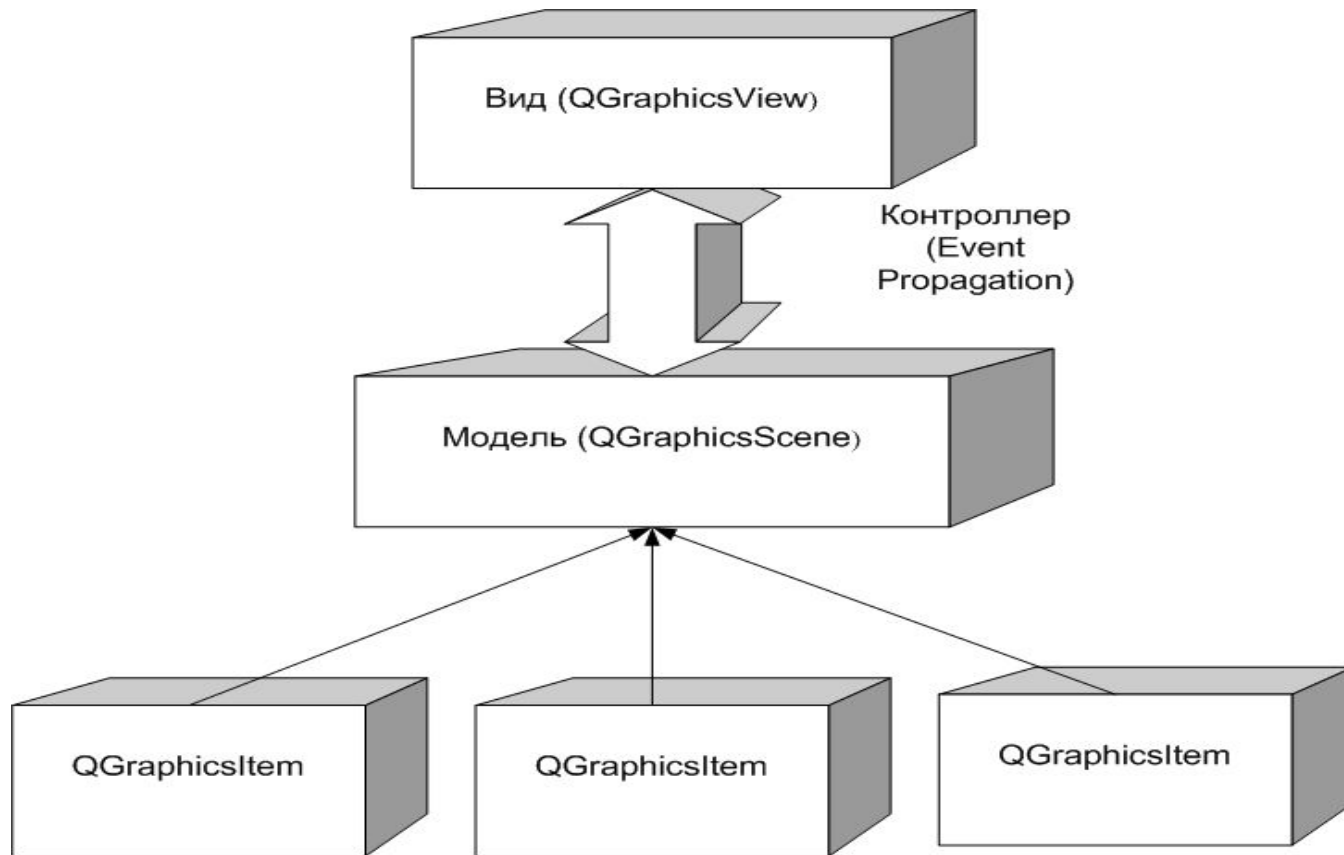
- http://www.opennet.ru/docs/RUS/qt3_prog/c4100.html



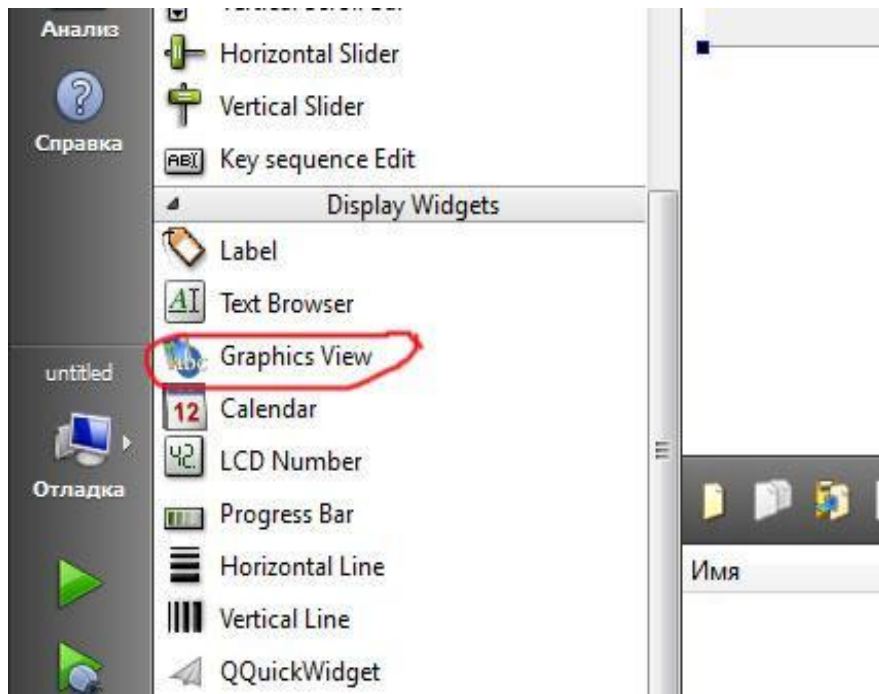
2D графика. QCanvas

- **QCanvas** - **QCanvasItem**
- *QCanvasLine, QCanvasRectangle, QCanvasPolygon, QCanvasPolygonalItem, QCanvasEllipse, QCanvasSpline, QCanvasSprite* и *QCanvasText..*
- ***QCanvasView***

http://www.opennet.ru/docs/RUS/qt3_prog/x4318.html



<http://qt-project.org/doc/qt-5/graphicsview.html>



addEllipse()
addText()
addLine()
addItem()
QGraphicsScene
QGraphicsView
setScene()

```
QGraphicsScene * scene = new QGraphicsScene;  
ui->graphicsView->setScene(scene);
```

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QGraphicsScene>
4 #include <QPixmap>
5 #include <QTextEdit>
6 #include <QGraphicsTextItem>
7 #include <QGraphicsWidget>
8
9 MainWindow::MainWindow(QWidget *parent) :
10     QMainWindow(parent),
11     ui(new Ui::MainWindow)
12 {
13     ui->setupUi(this);
14     QGraphicsScene * scene = new QGraphicsScene;
15     ui->graphicsView->setScene(scene);
16     scene->addPixmap(QPixmap("C:\\Qt\\MyProjects\\scene\\1.jpg"));
17     scene->addEllipse(QRectF(30,60, 300.0, 90.0));
18     scene->addText("Hello, world!");
```


addEllipse([qreal](#) x, [qreal](#) y, [qreal](#)w, [qreal](#) h, const [QPen](#) & pen = QPen(), const [QBrush](#) & brush = QBrush())

QRectF([qreal](#) x, [qreal](#) y, [qreal](#) width, [qreal](#) height)

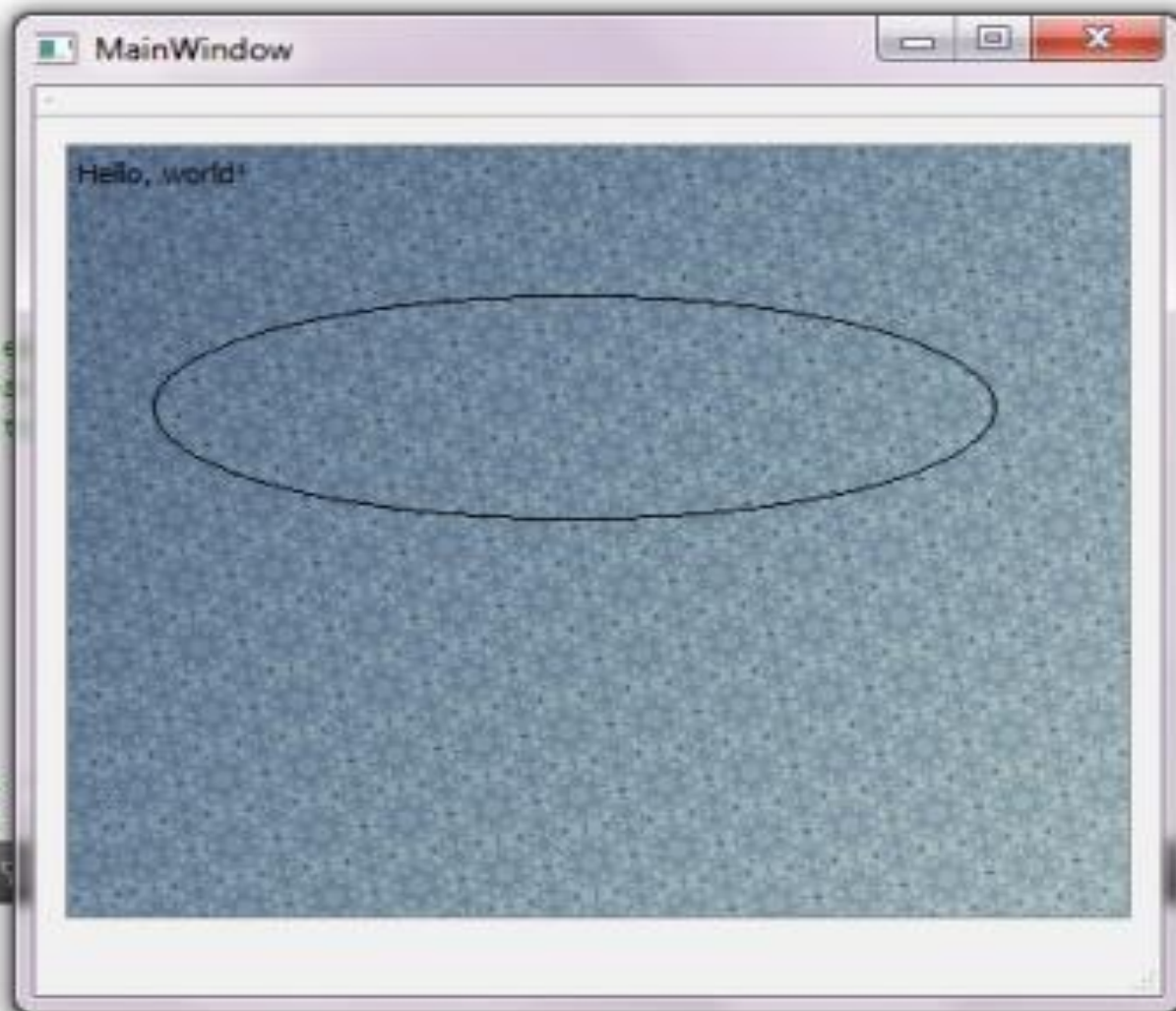
addText(const QString & text, const QFont & font = QFont())

addPixmap(const QPixmap & pixmap)

```
("Hello, world!");
```

```
tIt  
tPos  
em(t
```

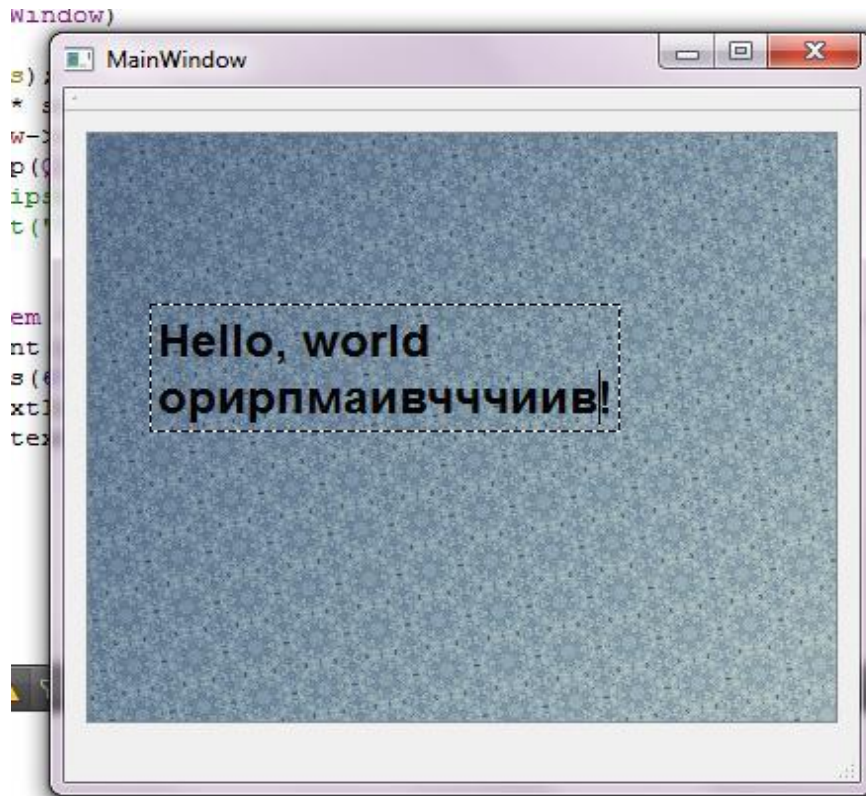
```
get  
get
```



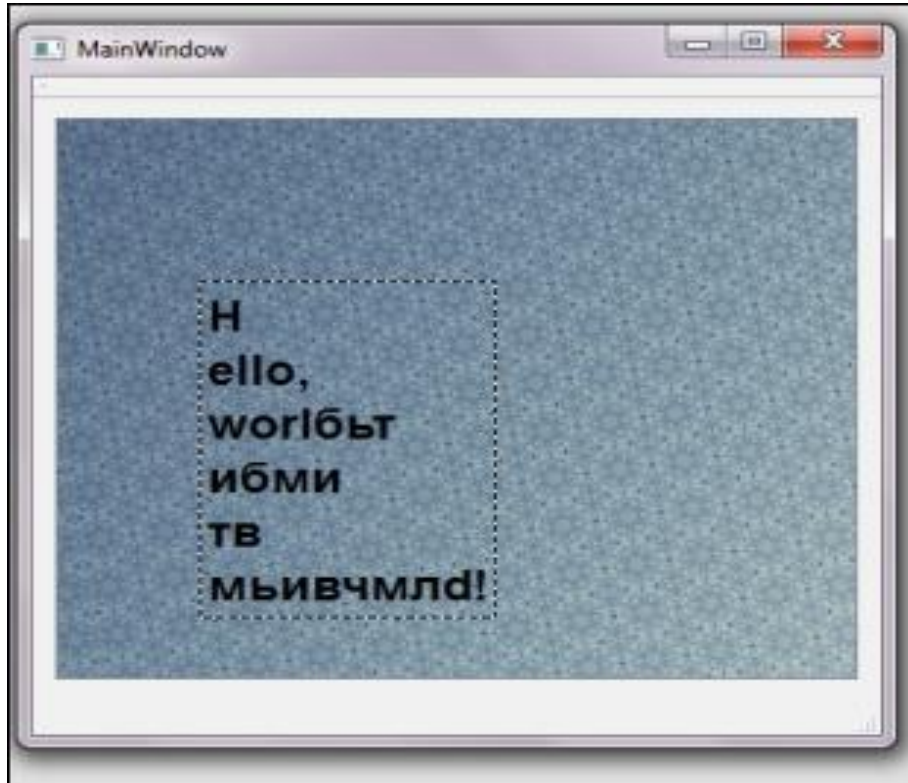
```
QGraphicsTextItem *textItem = new QGraphicsTextItem("Hello, world!");  
textItem->setFont(QFont("Times", 18, QFont::Bold));  
textItem->setPos(67, 90);  
scene->addItem(textItem);
```

Endow)





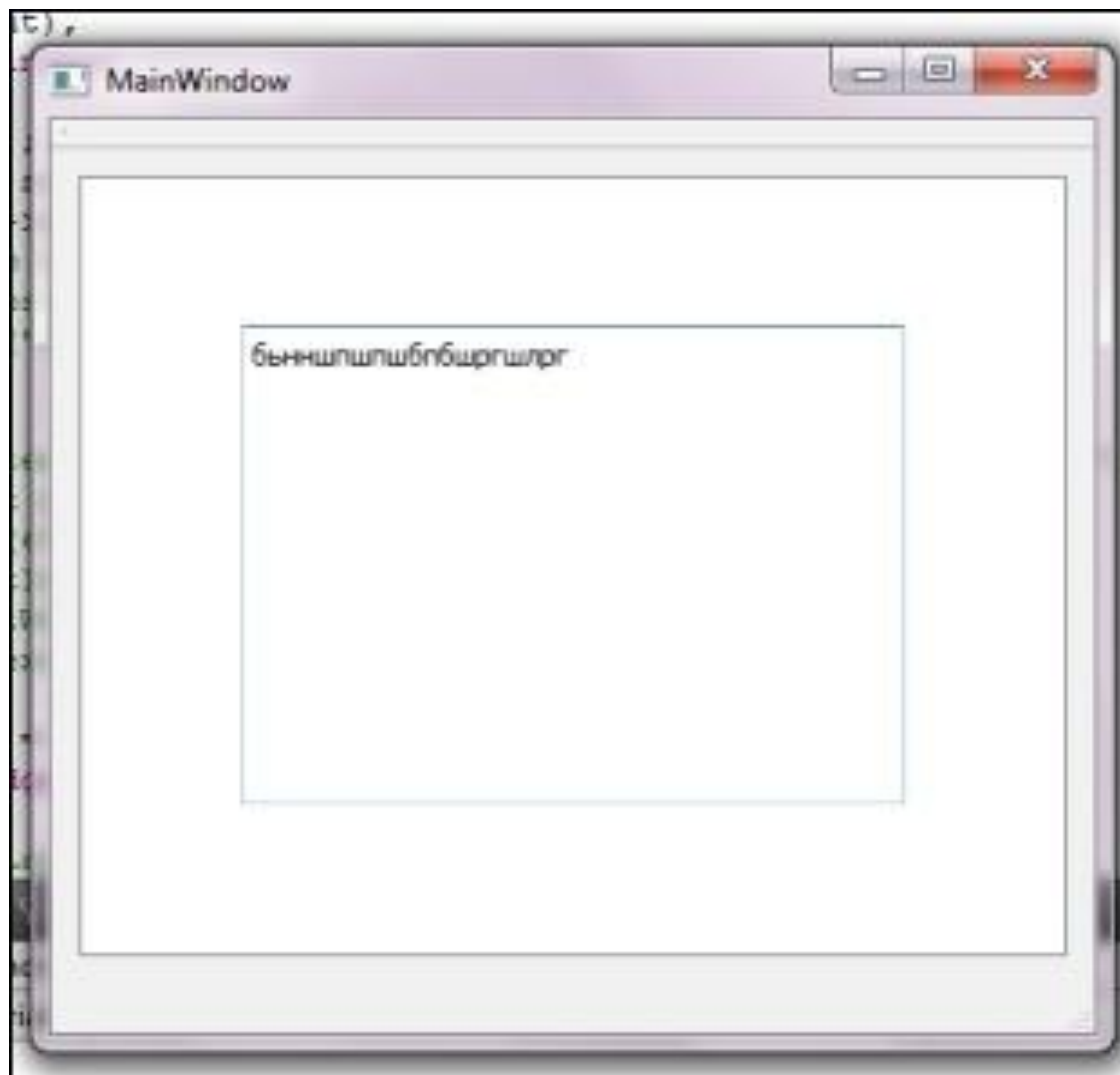
```
textItem->setTextInteractionFlags(Qt::TextEditable);
```

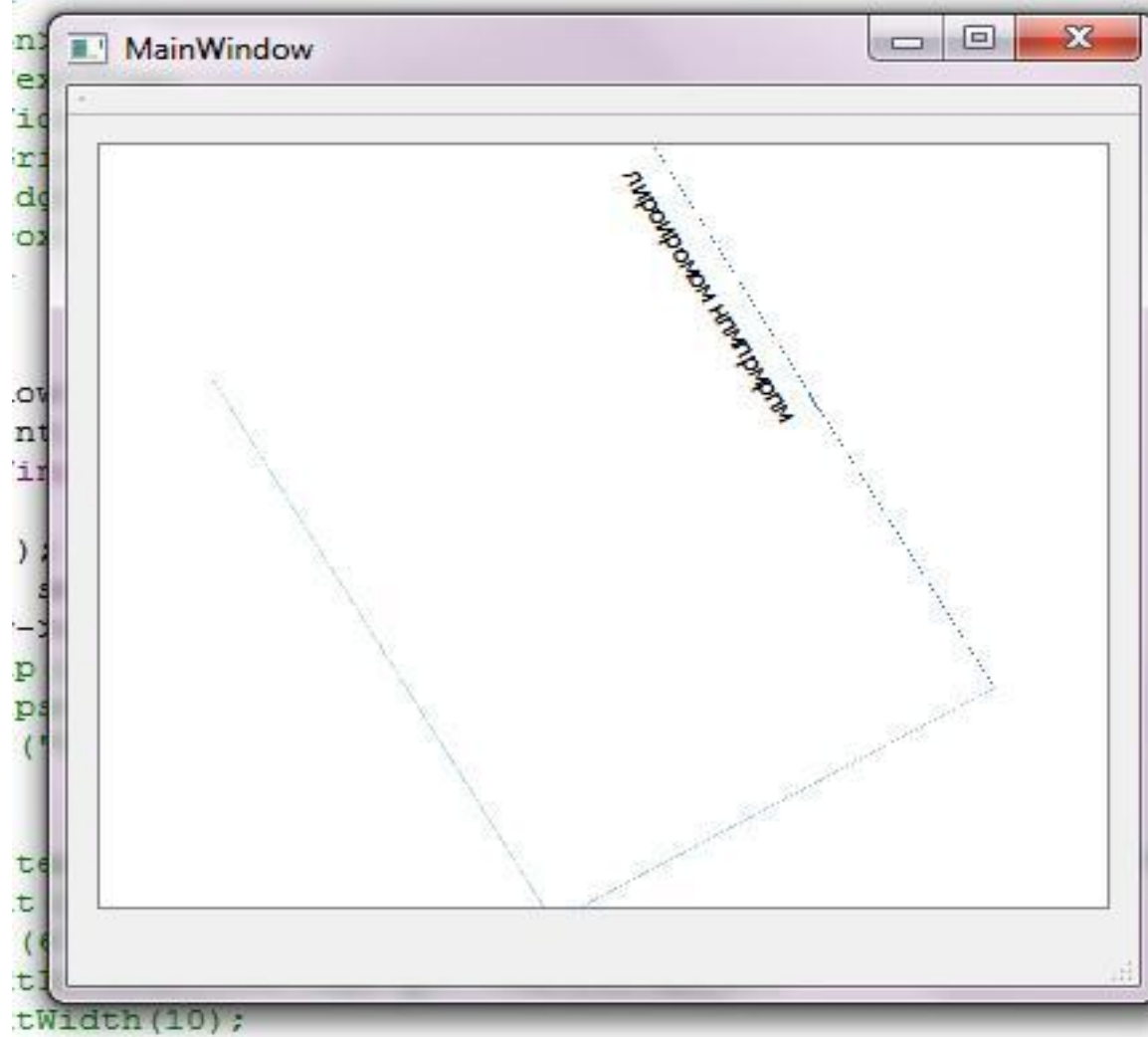


```
textItem->setTextWidth(10);
```

addWidget(QWidget * widget, Qt::WindowFlags wFlags = 0)

```
QTextEdit *tEdit = new QTextEdit;  
QGraphicsProxyWidget *proxy = scene->addWidget(tEdit);
```





setTransform()

```
tWidth(10);
```

```
QTextEdit *tEdit = new QTextEdit;
QGraphicsProxyWidget *proxy = scene->addWidget(tEdit);

QTransform transform = proxy->transform();
transform.translate(50., 30.);
transform.rotate(60.0);
proxy->setTransform(transform);
```

Game_Scene.h

```
9 class GameScene : public QGraphicsScene
10 {
11     Q_OBJECT
12 public:
13     GameScene(QObject *parent = 0);
14 protected:
15     virtual void keyPressEvent(QKeyEvent * keyEvent);
16 private:
17     QGraphicsPixmapItem * Cat;
18     QGraphicsPixmapItem * Dog;
19     QGraphicsPixmapItem * home;
20     int R;
21     void makeWalls();
22     void The_End();
23     void Win();
24     QGraphicsItem * itemCollidesWith(QGraphicsItem * item);
25
26 public slots:
27     void DogGo();
28     void Rand_dog();
29 };
```

<http://doc.crossplatform.ru/qt/4.6.x/qgraphicspixmapitem.html#details>

mainwindow.cpp

```
9
10 MainWindow::MainWindow(QWidget *parent) :
11     QMainWindow(parent),
12     ui(new Ui::MainWindow)
13 {
14     ui->setupUi(this);
15     GameScene * scene = new GameScene;
16     ui->graphicsView->setScene(scene);
17 }
18
19
```


Game_Scene.cpp *конструктор*

```
1  #include "Game_Scene.h"
2  #include <QGraphicsScene>
3  #include <QGraphicsTextItem>
4  #include <QGraphicsSceneMoveEvent>
5  #include <QKeyEvent>
6  #include <QGraphicsItem>
7  #include <QTransform>
8  #include <QMessageBox>
9  #include <QTime>
10 #include <QTimer>
11 #include <QDebug>
12
13 GameScene::GameScene(QObject *parent) : QGraphicsScene(parent)
14 {
15     makeWalls();
16     Cat = QPixmap("C:\\Qt\\MyProjects\\Labirint\\untitled\\black_cat.png");
17     QTransform tran = Cat->transform();
18     tran.translate(25.5, 25.5);
19     Cat->setTransform(tran);
20     Cat->setData(0, "Cat");
21 }
```

Game_Scene.cpp

конструктор

```
21  
22     home = QPixmap(QPixmap("C:\\Qt\\MyProjects\\Labirint\\untitled\\home.png"));  
23     tran = home->transform();  
24     tran.translate(555.5, 555.5);  
25     home->setTransform(tran);  
26     home->setData(0, "Home");  
27  
28     Dog = QPixmap(QPixmap("C:\\Qt\\MyProjects\\Labirint\\untitled\\dog.png"));  
29     tran = Dog->transform();  
30     tran.translate(260.5, 25.5);  
31     Dog->setTransform(tran);  
32     Dog->setData(0, "Dog");  
33
```

Game_Scene.cpp

конструктор

```
34 setBackgroundBrush(QBrush(QColor(255,255,255), QPixmap("C:\\Qt\\MyProjects\\Labirint\\untitled\\bg2.jpg")));
35
36 qsrand(QTime(0,0,0).secsTo(QTime::currentTime()));
37 Rand_dog();
38 QTimer * timer = new QTimer(this);
39 connect(timer, SIGNAL(timeout()), this, SLOT(DogGo()));
40 timer->start(100);
41
42 QTimer * timer2 = new QTimer(this);
43 connect(timer2, SIGNAL(timeout()), this, SLOT(Rand_dog()));
44 timer2->start(5000);
45
46 }
```


Game_Scene.cpp *makeWalls()*

```
85 void GameScene::makeWalls()
86 {
87     float walls[23][4] = {{0, 0, 20, 620},
88                           {20, 0, 620, 20},
89                           {625, 0, 20, 620},
90                           {20, 100, 620, 20},
91                           .....
92                           {25, 30, 150, 20},
93                           {400, 80, 20, 120},
94                           {525, 105, 20, 50}};
95
96     QBrush brush(QColor(255, 255, 255), QPixmap("C:\\Qt\\MyProjects\\Labirint\\untitled\\grass.jpg"));
97     QPen pen(Qt::NoPen);
98
99     for (int i = 0; i < 23; i++) {
100         QGraphicsItem * item = addRect(QRectF(walls[i][0], walls[i][1], walls[i][2], walls[i][3]), pen, brush);
101         item->setData(0, "Wall");
102     }
103 }
```

Game_Scene.cpp *keyPressEvent()*

```
48
49 void GameScene::keyPressEvent(QKeyEvent * keyEvent)
50 {
51     QPointF np;
52     np.setX(0);
53     np.setY(0);
54     switch (keyEvent->key()) {
55         case Qt::Key_Left:
56             np.setX(-10);
57             break;
58         case Qt::Key_Right:
59             np.setX(10);
60             break;
61         case Qt::Key_Up:
62             np.setY(-10);
63             break;
64         case Qt::Key_Down:
65             np.setY(10);
66             break;
67     }
68     QTransform tran = Cat->transform();
69     tran.translate(np.x(), np.y());
70     Cat->setTransform(tran);
```

Game_Scene.cpp *keyPressEvent()*

```
72     QGraphicsItem * obstacle = itemCollidesWith(Cat);
73     if (obstacle) {
74         if (obstacle->data(0) == "Wall") {
75             tran.translate(-np.x(), -np.y());
76             Cat->setTransform(tran);
77         }
78         else
79             if (obstacle->data(0) == "Dog") The_End();
80                 else if (obstacle->data(0) == "Home") Win();
81     }
82 }
83
```

Game_Scene.cpp

itemCollidesWith()

```
119 QGraphicsItem * GameScene::itemCollidesWith(QGraphicsItem * item)
120 {
121     QList<QGraphicsItem *> collisions = collidingItems(item);
122     foreach (QGraphicsItem * it, collisions) {
123         if (it == item)
124             continue;
125         return it;
126     }
127     return NULL;
128 }
```

Game_Scene.cpp *DogGo()*

```
140 void GameScene::DogGo ()
141 {
142     QPointF np;
143     np.setX(0);
144     np.setY(0);
145
146     switch (R)
147     {
148         case 1 : np.setX(-10); break; //left
149         case 2 : np.setX(+10); break; //right
150         case 3 : np.setY(-10); break; //top
151         case 4 : np.setY(+10); break; //bot
152     }
153
154     qDebug() << R;
155     QTransform tran = Dog->transform();
156     tran = Dog->transform();
157     tran.translate(np.x(), np.y());
158     Dog->setTransform(tran);
```

Game_Scene.cpp *DogGo()*

```
160 QGraphicsItem *obstacle = itemCollidesWith(Dog);
161 tran = Dog->transform();
162 if (obstacle) {
163     if ((obstacle->data(0) == "Wall" || (obstacle->data(0) == "Home"))) {
164         tran.translate(-np.x(), -np.y());
165         Dog->setTransform(tran);
166         Rand_dog();
167     }
168     else
169         if (obstacle->data(0) == "Cat") The_End();
170 }
171 }
```