Buổi 4 : Vẽ ảnh động

1. Hướng dẫn & bài tập

1.1. Hướng dẫn

Trong buổi 4, SV sẽ học cách sử dụng TimeEvent để vẽ các ảnh động. Thực chất ảnh động là một chuỗi các ảnh tĩnh được vẽ đi vẽ lại ở với tốc độ cao nên chúng ta mới có thể nhìn thấy vật chuyển động.

SV tạo một dự án mới, thiết kế giao diện vẫn như cũ, nhưng không dùng đến Groupbox và Button. Giao diện chương trình chỉ gồm 1 đối tượng là Widget được đặt tên là GraphicsControllers. Dự án sử dụng yếu tố liên quan đến thời gian nên các Button ít được dùng, vì khi dự án khởi động thì thời gian đã được tính, nên việc dùng các Button sẽ trở nên vô nghĩa; chỉ sử dụng đến Button khi ta cần gọi một hàm nào đó nằm ngoài sự kiện paintEvent của chương trình.

Trong file **graphics.h**, SV khai báo sự kiện timerEvent và các biến như sau:

```
void timerEvent(QTimerEvent *);
double position;
double timerId;
```

Biến position được sử dụng để làm tăng giá trị của một thuộc tính nào đó (vị trí, độ dài, …) khi thời gian tăng lên.

Biến timerId dùng để xác định khoảng thời gian lặp lại sự kiện vẽ, đơn vị tính là milisecond, 1s = 1000ms (vd: mỗi 10 ms sẽ vẽ lại 1 lần và cứ 10ms ta lại tăng biến position làm cho hình ảnh vật thay đổi)

Trong file **graphics.cpp**, SV khai báo nội hàm của sự kiện timeEvent:

```
void graphics::timerEvent(QTimerEvent *){
    position+=1;
    repaint();
}
```

Ở file **graphics.cpp**, trong hàm xây dựng của lớp **graphic**, khai báo mặc định cho biến position và timeId. Hàm xây dựng là **hàm được tạo sẵn** khi chúng ta tạo lớp graphics. SV chỉ thêm vào các dòng code sau:

```
#include "graphics.h"
#include <QPainter>

graphics::graphics(QWidget *parent) :
    QWidget(parent)
{
    position=0;
    timerId=startTimer(50);
}
```

Đến đây SV đã hoàn thành việc xây dựng khung một chương trình để vẽ ảnh động. Đoạn code timerId = startTimer(50) nghĩa là cứ mỗi 50 ms trôi qua, thì nó sẽ gọi sự kiện timerEvent() lên một lần. Mỗi lần sự kiện timerEvent được gọi lên thì chúng ta tăng giá trị của biến position lên 1 đơn vị, và ra lệnh repaint() để gọi sự kiện paintEvent() lên và vẽ lại hình ảnh.

Ví dụ 1: Khai báo một hàm drawExample. Hàm này sẽ vẽ một hình vuông và làm cho nó chuyển động theo hướng ngang (lưu ý tham số position trong hàm painter.drawRect).

```
void graphics::drawExample(QPainter& painter){
    int r=100;
    painter.setBrush(QBrush("#b40000"));
    painter.drawRect(position,h/2,r,r);
}
```

Gọi hàm này trong sự kiện paintEvent và thực thi chương trình

```
void graphics::paintEvent(QPaintEvent *){
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);
    drawExample(painter);
}
```

Nội dung file **graphics.h**:

```
#ifndef GRAPHICS_H
#define GRAPHICS_H

#include <QWidget>

class graphics : public QWidget
{
    Q_OBJECT
public:
    explicit graphics(QWidget *parent = 0);

    void timerEvent(QTimerEvent *);
    double position;
    double timerId;

    void paintEvent(QPaintEvent *);
    double h=height();
    double w=width();
    void drawExample(QPainter& painter);

signals:

public slots:

};

#endif // GRAPHICS_H
```

Nội dung file **graphics.cpp**:

```
#include "graphics.h"
#include <QPainter>

graphics::graphics(QWidget *parent) :
    QWidget(parent)
{
    position=0;
    timerId=startTimer(50);
}

void graphics::timerEvent(QTimerEvent *){
    position+=1;
    repaint();
}

void graphics::paintEvent(QPaintEvent *){
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);
    drawExample(painter);
}

void graphics::drawExample(QPainter& painter){
    int r=100;
    painter.setBrush(QBrush("#b40000"));
    painter.drawRect(position,h/2,r,r);
}
```

Sinh viên tự khảo khác các ví dụ khác:

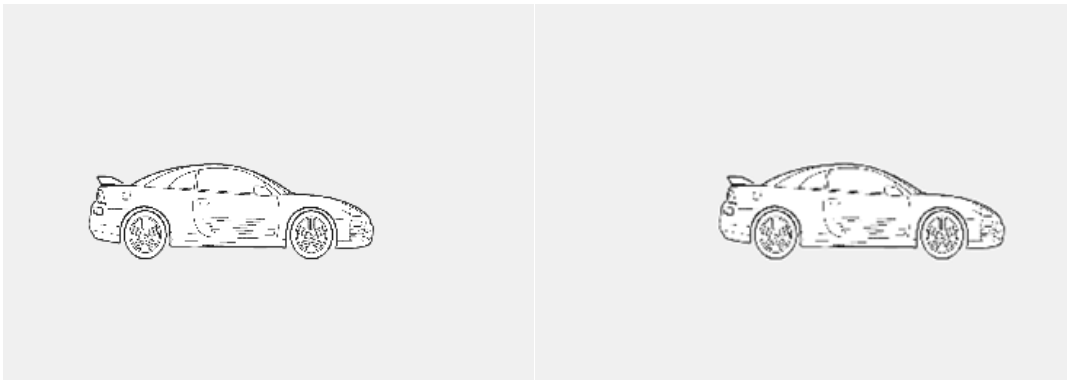
Ví dụ 2 : Khai báo một hàm drawExample. Hàm này sẽ vẽ một hình vuông và làm cho nó chuyển động theo hướng dọc (lưu ý tham số position trong hàm painter.drawRect).

Ví dụ 3 : Khai báo một hàm drawExample. Hàm này sẽ vẽ một hình vuông và làm cho nó xuất hiện theo hướng bất kỳ (lưu ý tham số position trong hàm painter.drawRect, sinh viên có thể thêm các biến khác phù hợp hơn).

1.2. Bài tập: SV có thể tuỳ ý sáng tạo và vẽ các hình động mới. Ví dụ: xe chạy thẳng, SV có thể thay đổi, xe chạy trên những con đường vòng vèo,…

1) Viết thủ tục để vẽ chiếc xe đang chạy

SV sử dụng hình ảnh oto.png (GV cung cấp) để vẽ. Dùng hàm drawPixmap() để vẽ các hình ảnh có sẵn. Hàm drawPixmap() nhận vào các tham số như hàm drawRect() và thêm vào 1 đối tượng QPixmap được tải lên từ đường dẫn tới file ảnh cần hiển thị.



Chiếc xe di chuyển nhanh hay chậm phụ thuộc vào việc chúng ta tăng/giảm biến position lên bao nhiêu sau mỗi timeEvent hoặc giảm/tăng thời gian startTimer(). Chúng ta cần cân đối 2 giá trị này để thu được kết quả tốt nhất.

Trong file **graphics.cpp** thêm #include <QString> và thay thế các dòng lệnh trong hàm **drawExample** như sau:

```
void graphics::drawExample(QPainter& painter){
    int r=100;
    int h=height();
    int w=width();
    painter.setBrush(QBrush("#b40000"));
   // painter.drawRect(position,h/2,r,r);

    QString filename = "D://oto.png";
    painter.drawPixmap(position,h/2,3*r,r,QPixmap(filename));

}
```

Lưu ý: SV phải đặt file ảnh oto.png đúng thư mục truy xuất.

****Sinh viên có thể sáng tạo bằng cách cho nhiều xe chạy, vẽ đường chạy khác cho xe, hoặc thêm nhiều xe khác,…. ****

2) Viết thủ tục để vẽ hình tròn chuyển động quanh quỹ đạo.

Khai báo thêm các biến và các hàm trong file graphics.h:

```
QPointF pstart,pcenter,pnew;
int angle;
QPointF quay(QPointF p, QPointF c, int deta);//quay diem p quanh diem c 1 goc deta (don vi do)
void animationCircle(QPainter& painter);
```

Khởi tạo và định nghĩa các hàm trong file graphics.cpp:

- Trong hàm xây dựng, khởi tạo biến angle=30 độ.

```
graphics::graphics(QWidget *parent) :
    QWidget(parent)
{
    position=0;
    timerId=startTimer(50);
    angle=30;
}
```

- Trong hàm timerEvent, tăng biến angle lên 30 độ.

```
void graphics::timerEvent(QTimerEvent *){
    position+=1;
    angle+=30;
    repaint();
}
```

- Trong hàm paintEvent thay thế hàm drawExample thành hàm animationCircle.

```
void graphics::paintEvent(QPaintEvent *){
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);
   // drawExample(painter);
    animationCircle(painter);
}
```

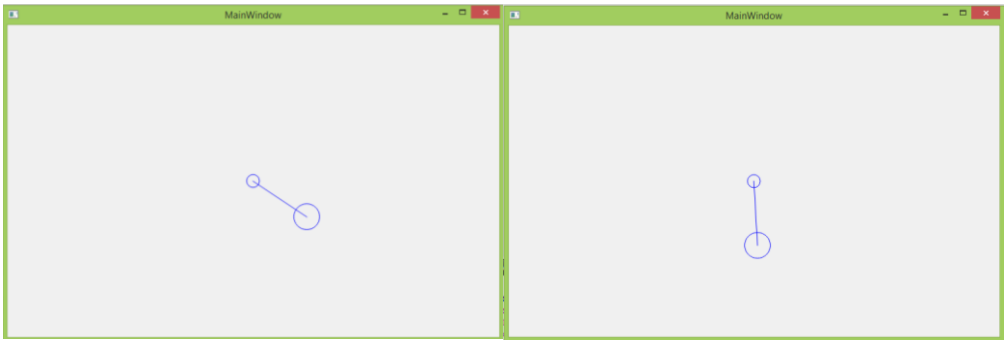
- Hàm quay được định nghĩa như sau:

```
QPointF graphics::quay(QPointF p, QPointF c, int deta){//quay điểm p quanh điểm bất kỳ c, 1 góc deta (độ)
    QPointF pnew;
    double goc=deta*3.14/180; // đổi độ về radian
    pnew.setX(c.x()+(p.x()-c.x())*cos(goc)-(p.y()-c.y())*sin(goc));
    pnew.setY(c.y()+(p.x()-c.x())*sin(goc)+(p.y()-c.y())*cos(goc));
    return pnew;
}
```

- Hàm animationCircle được định nghĩa như sau:

```
void graphics::animationCircle(QPainter& painter){
    int r=10;
    int h=height()/2;
    int w=width()/2;
    pcenter.setX(w);
    pcenter.setY(h);
    pstart.setX(pcenter.x());
    pstart.setY(pcenter.y()-100);

    pnew=quay(pstart,pcenter,angle);
    painter.setPen(Qt::blue);
    painter.drawEllipse(pcenter,r,r);
    painter.drawLine(pcenter,pnew);
    painter.drawEllipse(pnew,2*r,2*r);

}
```

Chạy chương trình:



3) Viết thủ tục để vẽ ngôi sao nội tiếp hình tròn và chuyển động quanh quỹ đạo (cách thực hiện tương tự như bài tập 2).

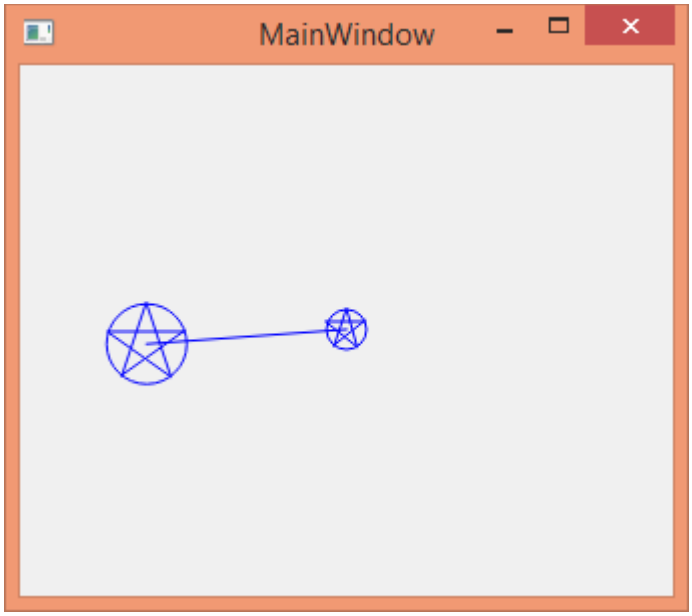
SV bổ sung 2 dòng lệnh sau vào hàm animaionCircle và định nghĩa hàm vẽ ngôi sao drawStar (SV có thể viết cách khác)

```
void graphics::animationCircle(QPainter& painter){
    int r=10;
    int h=height()/2;
    int w=width()/2;
    pcenter.setX(w);
    pcenter.setY(h);
    pstart.setX(pcenter.x());
    pstart.setY(pcenter.y()-100);

    pnew=quay(pstart,pcenter,angle);
    painter.setPen(Qt::blue);
    painter.drawEllipse(pcenter,r,r);
    painter.drawLine(pcenter,pnew);
    painter.drawEllipse(pnew,2*r,2*r);

    drawStar(painter,pcenter,r);
    drawStar(painter,pnew,2*r);

}
```

Định nghĩa hàm drawStar

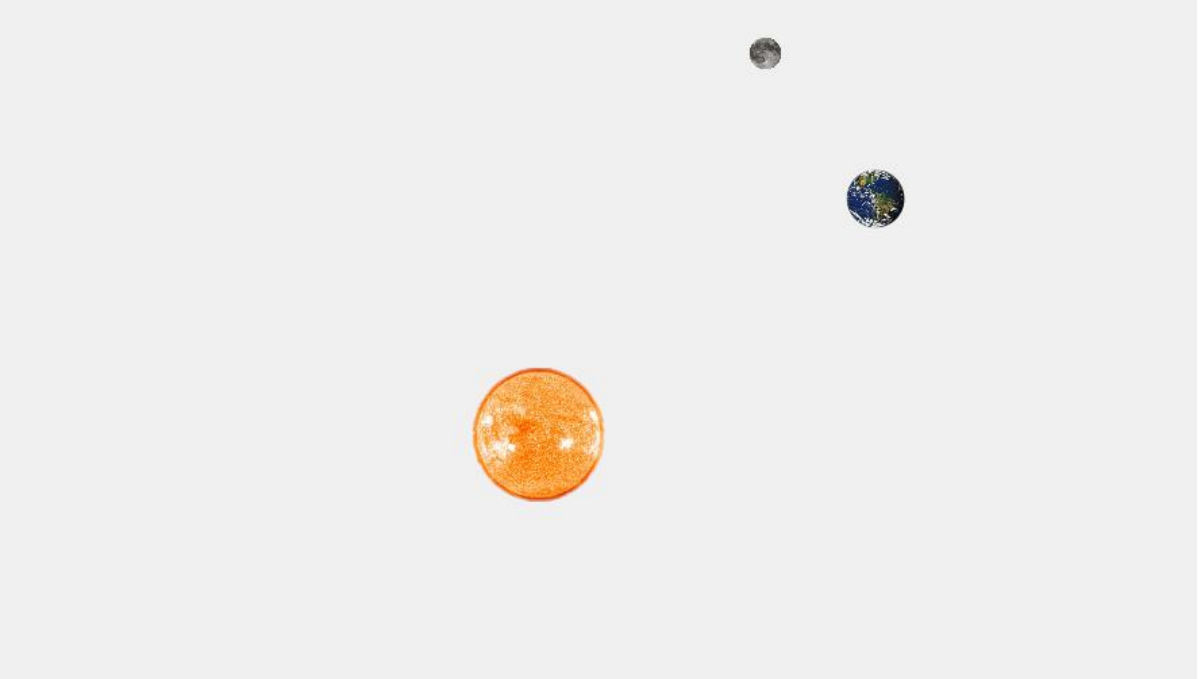
```
void graphics::drawStar(QPainter& painter,QPointF pcenter,int r){
    QPointF p(pcenter.x(),pcenter.y()-r); //p điểm bắt đầu
    QPolygon polygon;
    int goc=72;
    polygon<<QPoint(p.x(),p.y());
    for (int i=1;i<=4;i++){
        p=quay(p,pcenter,goc);
        polygon<<QPoint(p.x(),p.y());
    }
    painter.drawLine(polygon.value(0),polygon.value(2));
    painter.drawLine(polygon.value(0),polygon.value(3));
    painter.drawLine(polygon.value(1),polygon.value(3));
    painter.drawLine(polygon.value(1),polygon.value(4));
    painter.drawLine(polygon.value(2),polygon.value(4));
}
```

Chạy chương trình:



2. Bài tập nâng cao:

1) **Viết thủ tục để vẽ hệ mặt trời**
   SV sử dụng các file ảnh *moon.png, earth.png và sun.png* do GV cung cấp
   Yêu cầu của bài tập :
   - Mặt trời được vẽ ở giữa màn hình
   - Trái đất chuyển động xung quanh mặt trời theo quỹ đạo hình tròn.
   - Mặt trăng chuyển động xung quanh trái đất theo quỹ đạo hình tròn với vận tốc lớn hơn vận tốc của trái đất khi chuyển động quanh mặt trời (các vận tốc chỉ mang tính tương đối).



2) **Viết thủ tục để vẽ 1000 ngôi sao** (có màu nhiều màu khác nhau) xuất hiện/chuyển động ngẫu nhiên trên bầu trời theo thời gian.

**** SV có thể thêm các hành tinh khác trong thái dương hệ vào để có được những ảnh mới, và thực hiện các ý tưởng khác với các đối tượng ảnh cơ bản đã được vẽ trong những bài tập ở những buổi trước ****