Môn học : Kỹ thuật đồ họa

Tài liệu hướng dẫn thực hành

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 ta mới có thể nhìn thấy vật chuyển động.

Sinh viên 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 có 1 đối tượng là Widget được đặt tên là Graphics Controllers mà thôi. Do chúng ta sử dụng đến yếu tố thời gian trong dự án này nên ta sẽ hạn chế sử dụng các button. Vì Khi dự án khởi đông 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**, sinh viên khai báo sự kiện time event và các biến như sau :

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

Ta sẽ sử dụng biến position để làm tăng giá trị của một thuộc tính nào đó (vị trí, độ dài, v…v…) khi thời gian tăng lên. Còn biến timeId 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**, ta tiến hành khai báo nội hàm của sự kiện timeEvent :

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

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

```
#include <math.h>
#include <QString>
#include <QList>

graphics::graphics(QWidget *parent) :
    QWidget(parent)
{

    position = 0;
    timerId = startTimer(50);


}
```

Đến đây các bạn đã hoàn thành việc xây dựng khung một chương trình để vẽ ảnh động. Đoạn code timerId = startTimer(50) có nghĩa là cứ mỗi 50 ms trôi qua, thì nó sẽ gọi sự kiện timeEvent() lên một lần. MỖi lần sự kiện timeEvent được gọi lên thì 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 của chúng ta.

Ví dụ :

Khai báo một hàm drawExample như sau. Hàm này sẽ vẽ một hình vuông và làm cho nó chuyển động theo hướng ngang.

```
void graphics::drawExample(QPainter &painter)
{
    /* ...*/
    h = height();
    w = width();
    painter.setBrush(QBrush("#4b4848"));
    painter.drawRect(position,h/2, 50,50);
}
```

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

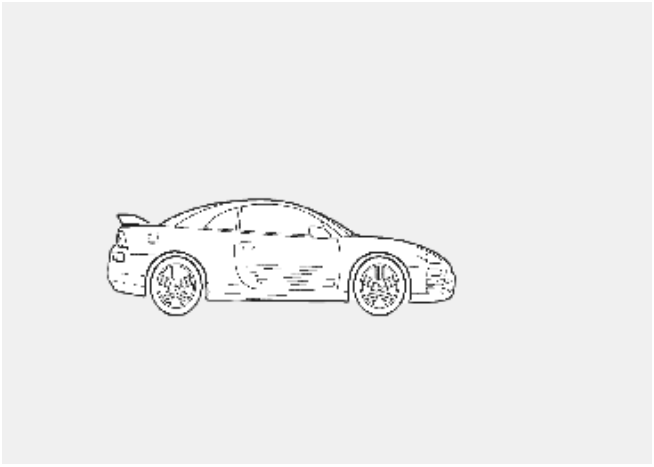
```
void graphics::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    drawExample(painter);

}
```

1.2. Bài tập

*****Lưu Ý *****

Các bài tập trong buổi 4 không ràng buộc sinh viên phải làm giống y như mẫu của giáo viên. SV có thể tùy ý sáng tạo và vẽ những hình động mới dựa trên bài tập giáo viên ra. VD : Thay vì cho xe chạy thẳng, SV có thể vẽ đường chạy vòng vèo cho xe, hoặc cho nhiều xe chạy qua lại, v…v…Sự sáng tạo càng lớn, SV càng có khả năng được điểm cộng (ở tất cả các bài chứ không riêng gì bài tập nâng cao).

1) Viết thủ tục để vẽ chiếc xe đang chạy
   SV sử dụng hình ảnh oto.png mà giáo viên cung cấp để vẽ. Dùng hàm drawPixmap() để vẽ các hình ảnh có sẵn.
   Hàm drawPixmap() nhận vào các thông số như hàm drawRect(), chỉ thêm vào 1 đối tượng Qpixmap được load lên từ đường dẫn tới file ảnh mà chúng ta muốn hiển thị.



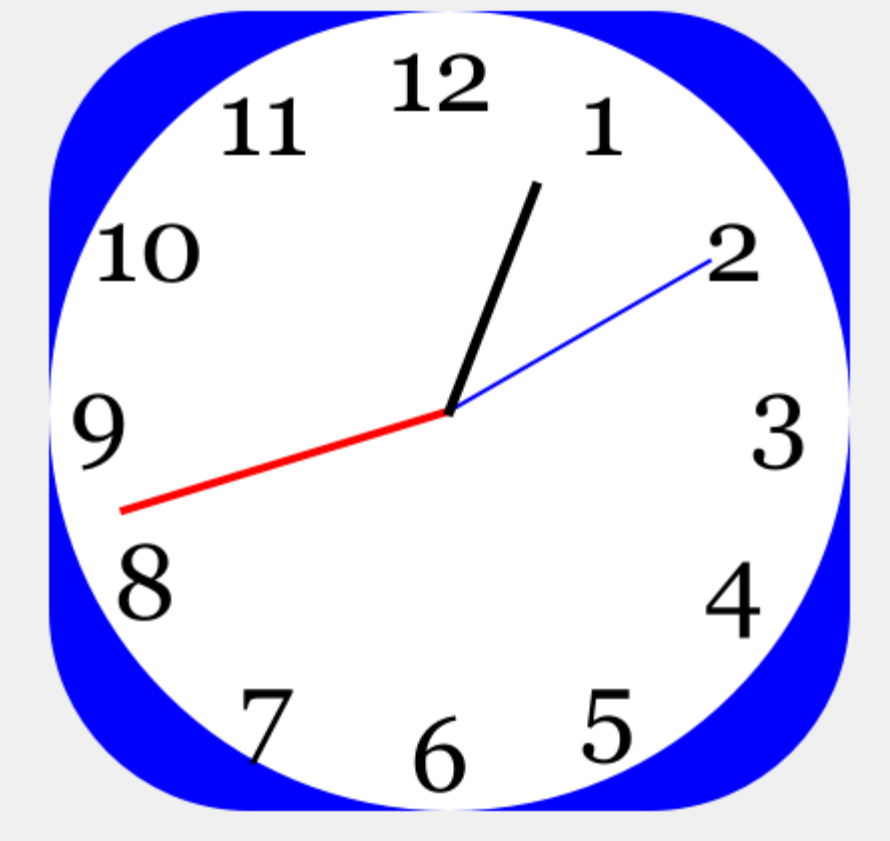
Việc chiến xe di chuyển nhanh hay chậm phụ thuộc vào việc chúng ta tăng biến position lên bao nhiêu sau mỗi timeEvent, hoặc giảm thời gian startTimer(). Cần cân đối 2 giá trị này để thu được kết quả tốt nhất.

****Sinh viên có thể sáng tạo bằng cách cho nhiều xe chạy, vẽ đường chạy ngoằn ngoèo cho xe, hoặc thêm nhiều xe khác vào, tổ chức thành một cuộc đua xe, v…v…. ****

Sẽ có điểm cộng đối với những sáng tạo thú vị

2) Viết thủ tục để vẽ đồng hồ treo tường, có kim giờ, kim phút, và kim giây.

****Trong bài tập này, điểm sáng tạo sẽ dành cho phần thẩm mĩ. Sinh viên tự do sáng tạo giao diện của đồng hồ, kim giờ, kim phút, kim giây ****



SV dùng QPainterPath và hàm painter.drawPath() để vẽ các con số. Tô chọn các hàm này rồi bấm F1 để xem các hướng dẫn sử dụng.

Kim giây di chuyển một vòng 360 hết 60s, vậy trong 1s kim giây sẽ dịch chuyển 1 góc là 360/60 = 6 độ. Chia cung tròn ra làm một đa giác đều có 60 cạnh, dùng công thức sau để tính tọa độ mỗi cạnh

- X = R Cos(2 * pi * i / n)
- Y = R Sin(2 * pi * i / n)

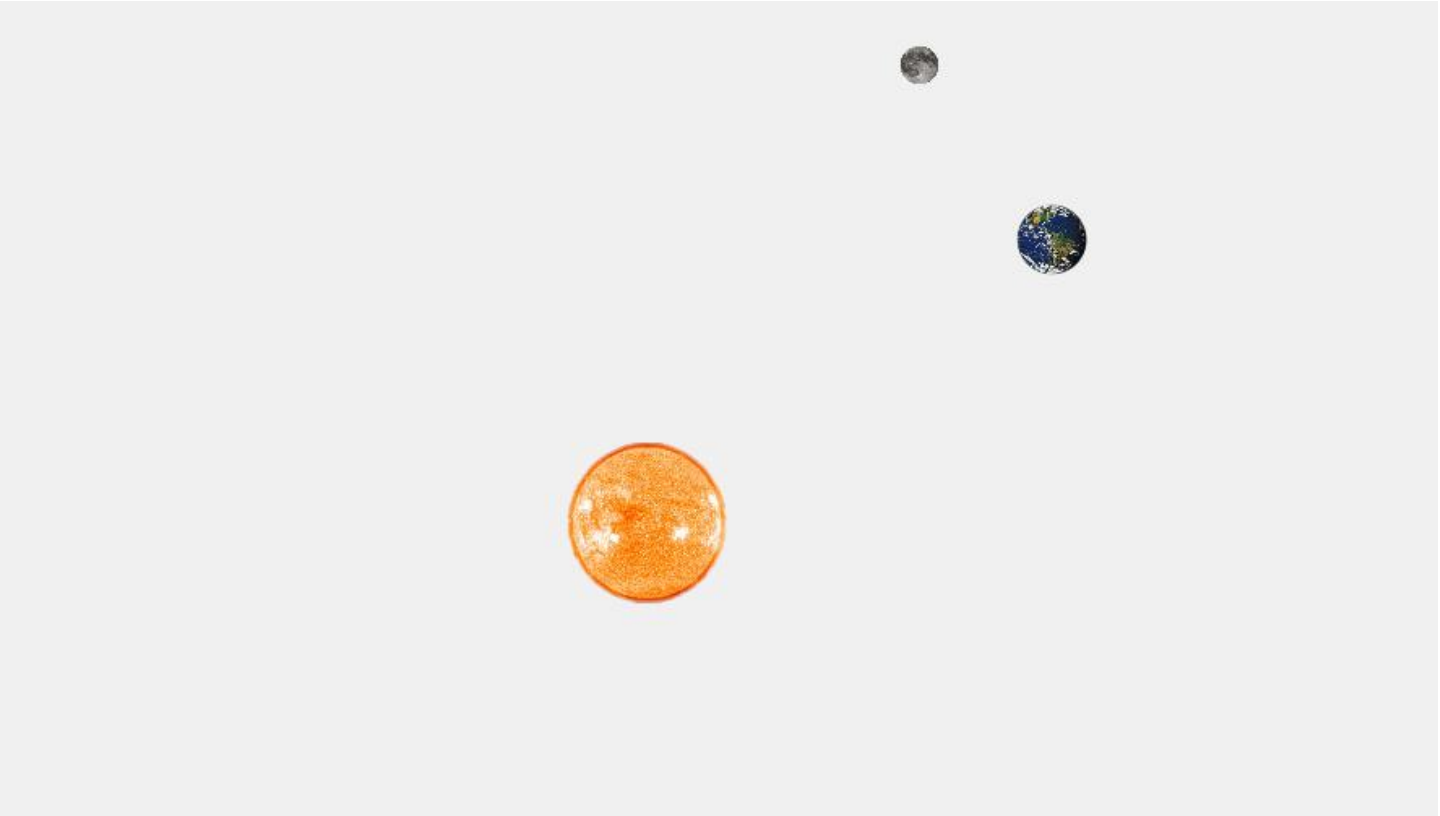
Trong trường hợp này, n = 60 và i = 0…. n-1
Tương tự đối với kim giờ và kim phút.

3) Viết thủ tục để vẽ hệ mặt trời

   Trong bài tập này chỉ giới hạn ở việc vẽ mặt trời, trái đất và mặt trăng. SV sử dụng các file ảnh *moon.png, earth.png và sun.png* do giáo viên cung cấp để vẽ hình các đối tượng này.

   Yêu cầu của bài tập :

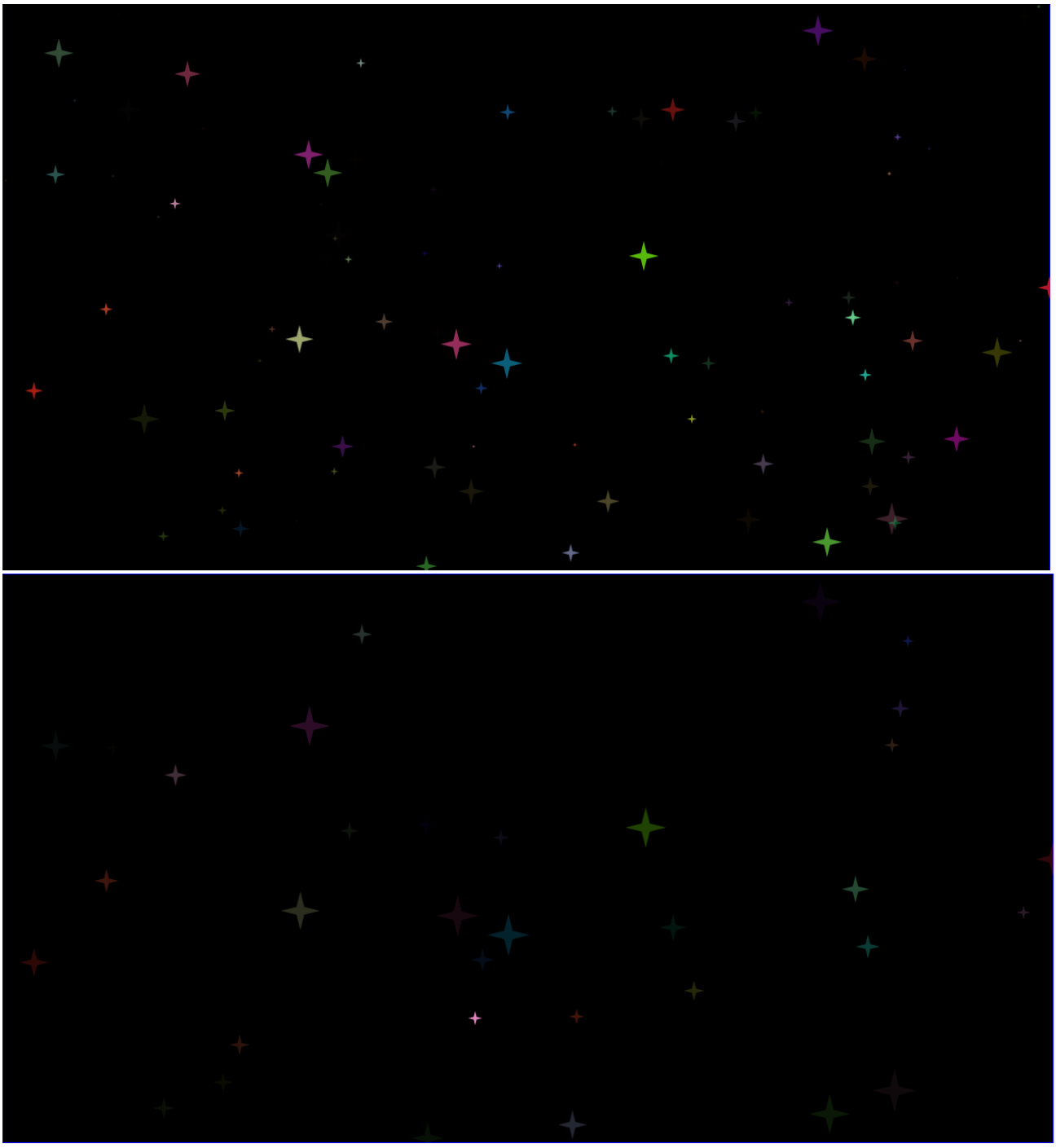
   - Mặt trời được vẽ ở giữa màn hình
   - Trái đất chuyển động xung quanh mặt trời theo quỹ đạo hình tròn.
   - Mặt trăng chuyển động xung quanh trái đất theo quỹ đạo hình tròn với vận tốc lơn hơn vận tốc của trái đất khi chuyển động quanh mặt trời. (các vận tốc chỉ mang tính tương đối).



**** SV có thể thêm các hành tinh khác trong thái dương hệ vào để có thể có điểm sáng tạo ở bài tập này ****

4) **Bài tập nâng cao : SV làm đúng và giải thích đúng sẽ được cộng 0.5đ vào điểm thi cuối kỳ.**

Viết thủ tục để vẽ bầu trời sao lấp lánh



Yêu cầu :

- Các ngôi sao được vẽ bởi một màu khác nhau
- Các ngôi sao giữ nguyên vị trí (không thay đổi tọa độ) sau mỗi sự kiện timeEvent và paintEvent
- Sau các sự kiện timeEvent, thay đổi kích thước của từng ngôi sao (tăng lên và sau đó giảm đi)
- Sau các sự kiện timeEvent, thay đổi độ trong suốt (opacity) của từng ngôi sao (các ngôi sao hiện rõ và sau đó mờ dần đi)