BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

#   



**BÁO CÁO**

**MÔN ĐỒ HỌA MÁY TÍNH**

Đề tài

ỨNG DỤNG ĐỒ HỌA MÁY TÍNH

VÀO VIỆC VẼ PHONG CẢNH

**Giáo viên hướng dẫn:**

***Ths. Phạm Xuân Hiền***

**Sinh viên thực hiện:**

**Đỗ Thành Công B1709526**

**Nguyễn Hưng B1709536**

NHẬN XÉT CỦA GIẢNG VIÊN

LỜI CẢM ƠN

Đầu tiên tôi xin gửi lời cảm ơn sâu sắc nhất tới cô Ths.Phạm Xuân Hiền. Cô đã hướng dẫn, tận tình chỉ bảo, giúp đỡ chúng em trong suốt quá trình học môn học này.

*Cần Thơ*, ngày 27 tháng 12 năm 2020

Người viết

**Đỗ Thành Công**

MỤC LỤC

[CHƯƠNG 1: GIỚI THIỆU 1](#_Toc59999379)

[1. Đặt vấn đề: 1](#_Toc59999380)

[2. Mục tiêu đề tài: 1](#_Toc59999381)

[3. Đối tượng và phạm vi nghiên cứu 1](#_Toc59999382)

[4. Phương pháp nghiên cứu 1](#_Toc59999383)

[5. Kết quả đạt được 1](#_Toc59999384)

[6. Bố cục đồ án 1](#_Toc59999385)

[CHƯƠNG 2: MÔ TẢ BÀI TOÁN 2](#_Toc59999386)

[1. Mô tả chi tiết bài toán: 2](#_Toc59999387)

[2. Vấn đề liên quan đến bài toán 2](#_Toc59999388)

[3. Giải pháp cho bài toán 2](#_Toc59999389)

[CHƯƠNG 3 : THIẾT KẾ VÀ CÀI ĐẶT 3](#_Toc59999390)

[1. Thiết kế hệ thống 3](#_Toc59999391)

[2. Thiết kế và cài đặt giải thuật 4](#_Toc59999392)

[3. Kết quả. 12](#_Toc59999393)

[CHƯƠNG 4 : KẾT QUẢ ĐẠT ĐƯỢC 14](#_Toc59999394)

[TÀI LIỆU THAM KHẢO 15](#_Toc59999395)

DANH MỤC HÌNH

[Hình 1: Khung cảnh vào ban ngày 13](#_Toc59999427)

[Hình 2: Khung cảnh vào ban đê 13](#_Toc59999428)

DANH MỤC BẢNG

[Bảng 1: Các biến toàn cục 3](#_Toc59999492)

[Bảng 2: Các hàm sử dụng 3](#_Toc59999493)

# CHƯƠNG 1: GIỚI THIỆU

## Đặt vấn đề:

Việc áp dụng các kĩ thuật Đồ họa máy tính vào việc vẽ phong cảnh đã được sử dụng phổ biến trong nhiều năm nay, sau khi tìm hiểu và nghiên cứu nhóm chúng em quyết định sử dụng các kĩ thuật trong Đồ họa máy tính để vẽ cảnh.

## Mục tiêu đề tài:

Sử dụng các kĩ thuật Đồ họa máy tính và các đối tượng cơ sở để vẽ cảnh.

## Đối tượng và phạm vi nghiên cứu

* Các đối tượng Đồ họa cơ sở
* Phần mềm QT Creator.
* Các kĩ thuật vẽ trong QT Creator.

## Phương pháp nghiên cứu

* Phân tích
* Thiết kế
* Thực hiện

## Kết quả đạt được

Đã xây dựng được chương trình thỏa mãn mục tiêu của đề tài.

## Bố cục đồ án

Chương 1: Giới thiệu.

Chương 2: Mô tả bài toán.

Chương 3: Thiết kế và cài đặt.

Chương 4: Kết quả đạt được.

# CHƯƠNG 2: MÔ TẢ BÀI TOÁN

## Mô tả chi tiết bài toán:

* Sử dụng các hình ảnh như núi, mặt trăng, chim có sẵn.
* Xây dựng các hàm đễ vẽ thuyền, mặt trời, biển, mây, sao, cờ và background.
* Xây dựng các hàm xử lý việc di chuyển cũng như thay đổi vị trí của các đối tượng.

## Vấn đề liên quan đến bài toán

* Xử lý việc di chuyển của mặt trăng và mặt trời.
* Xử lý việc di chuyển của mây, chim và thuyền.

## Giải pháp cho bài toán

* Tính toán tâm quay, góc quay của mặt trăng và mặt trời.
* Tính toán tỉ lệ thành phần của chiếc thuyền.
* Tính toán tỉ lệ và góc quay của các cánh sao.
* Tính toán vị trí di chuyển của mây, chim và thuyền.

# CHƯƠNG 3 : THIẾT KẾ VÀ CÀI ĐẶT

## Thiết kế hệ thống

Khai báo các biến toàn cục và các hàm sử dụng trong Header: Graphics.h

|  |  |
| --- | --- |
| Tên biến | Ý nghĩa |
| timerId | Xác đinh khoảng thời gian lặp lại sự kiện vẽ |
| moonAngle | Góc quay của mặt trăng |
| sunAngle | Góc quay của mặt trời |
| luminosity | Độ tỏa sáng của mặt trời |
| speedOfBoat | Tốc độ di chuyển của thuyền |
| speedOfCloud1 | Tốc độ di chuyển của đám mây thứ nhất |
| speedOfCloud2 | Tốc độ di chuyển của đám mây thứ hai |
| speedOfBird | Tốc độ di chuyển của chim |

Bảng 1: Các biến toàn cục

|  |  |
| --- | --- |
| Tên hàm | Ý nghĩa |
| paintEvent | Sự kiện vẽ |
| timerEvent | Hàm sẽ được gọi lại sau một khoảng thời gian timerId |
| quay | Quay điểm p quay điểm c một góc deta |
| trungdiem | Trả về trung điểm của a và b |
| random | Trả về một số ngẫu nhiên |
| randomNM | Trả về một số ngẫu nhiên trong khoảng N – M |
| backGround | Xác định màu nền |
| mountain | Vẽ núi |
| sea | Vẽ biển |
| moon | Vẽ mặt trăng |
| bird | Vẽ chim |
| thaiCuc | Vẽ thái cực |
| sun | Vẽ mặt trời |
| boat | Vẽ thuyền |
| cloud | Vẽ mây |
| star | Vẽ sao |

Bảng 2: Các hàm sử dụng

## Thiết kế và cài đặt giải thuật

* Phương thức lớp graphics:

|  |
| --- |
| graphics::graphics(QWidget \*parent) :  QWidget(parent)  {  timerId=startTimer(250); Sau 250ms thì gọi lại hàm timerEvent  luminosity=0; Độ lan tỏa ánh sáng của mặt trời ban đầu là 0  sunAngle =0; Góc quay ban đầu của mặt trời so với tâm là 0  moonAngle =180; Góc quay ban đầu của mặt trăng so với tâm là 180  speedOfBoat=0; Tốc độ ban đầu của thuyền là 0  speedOfBird=0; Tốc độ ban đầu của chim là 0  } |

* Phương thức timerEvent:

|  |
| --- |
| void graphics::*timerEvent*(QTimerEvent \*){  speedOfBoat+=5;  speedOfBird +=10;  speedOfCloud1 +=10;  speedOfCloud2 +=8;  sunAngle +=4;  moonAngle-=4;  if(luminosity==30\*45){  luminosity = 0;  }  else{  luminosity +=30;  }  repaint();  } |

Sau mỗi 250ms thì:

* Vị trí của thuyền tăng thêm 5px so với vị trị trước đó.
* Vị trí của chim tăng thêm 10px so với vị trí trước đó.
* Vị trí của đám mây thứ nhất tăng thêm 10px so với vị trí trước đó.
* Ví trí của đám mây thứ hai tăng thêm 8px so với vị trí trước đó.
* Góc quay của mặt trời tăng thêm 4 độ so với vị trí trước đó.
* Góc quay của mặt trăng giảm 4 độ so với vị trí trước đó.
* Khi mặt trời quay đến 180 độ thì đặt lại độ lan tỏa bằng 0.
* repaint(): gọi lại hàm paintEvent.
* Phương thức paintEvent:

|  |
| --- |
| void graphics::*paintEvent*(QPaintEvent \*){  QPainter painter(this);  painter.setRenderHint(QPainter::Antialiasing);  backGround(painter);  for(int i=1;i<=1000;i++){  int w =370;  QPoint p(i+1,random(w));  star(painter,p,2);  i+=10;  }  painter.setBrush(QBrush("#8787ff"));  QPoint sunP(0,height());  QPoint O(width()/2,height());  sunP = quay(sunP,O,sunAngle);  sun(painter,sunP,50);  sea(painter);  QPoint cloud1(300,130);  cloud1.setX(cloud1.x()+ speedOfCloud1);  QPoint cloud2(650,200);  QPoint cloud3(450,50);  QPoint cloud4(280,130);  QPoint cloud5(50,50);  QPoint cloud6(60,70);  cloud6.setX(cloud6.x()+ speedOfCloud2);  cloud(painter,cloud1,40);  cloud(painter,cloud2,40);  cloud(painter,cloud3,60);  cloud(painter,cloud4,35);  cloud(painter,cloud5,40);  cloud(painter,cloud6,50);  mountain(painter);  QPoint boatP(0,370);  boatP.setX(boatP.x()+ speedOfBoat);  boat(painter,boatP,150);  QPoint birdP(100,200);  birdP.setX(birdP.x()+ speedOfBird);  bird(painter,birdP);  } |

* Gọi hàm background đề vẽ màu nền.
* Vòng lặp for để vẽ 100 ngôi sao trên bầu trời với màu ngẫu nhiên.
* Điểm sunP để xác định vị trí ban đầu của mặt trời
* Điểm O xác định tâm quay của mặt trời.
* Quay mặt trời quanh tâm O với góc sunAngle đã được khai báo ở timerEvent.
* Vẽ mặt trời
* Vẽ biển.
* Xác định các vị trí của các đám mây.
* Di chuyển các đám mây theo trục x với speedOfCloud1 và speedOfCloud2 được khai báo ở timerEvent.
* Vẽ núi.
* Xác định vị trí của thuyền.
* Di chuyển thuyền theo trục x với speedOfBoat được khai báo ở timerEvent.
* Vẽ thuyền.
* Xác định vị trí của chim.
* Di chuyển vị trí chim theo trục x với speedOfBird được khai báo ở timerEvent.
* Vẽ chim.
* Các phương thức cơ sở:

|  |
| --- |
| QPoint graphics::quay(QPoint p,QPoint c,int deta){  QPoint pnew;  double goc=deta\*3.14/180;  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;  }  QPoint graphics::trungdiem(QPoint a, QPoint b){  QPoint pnew((a.x()+b.x())/2, (a.y()+b.y())/2);  return pnew;  } |

* Các phương thức trả về số ngẫu nhiên:

|  |
| --- |
| int graphics::random(int n)  {  return rand() % n;  }  int graphics::randomNM(int n,int m){  return rand()% m+n;  } |

* Phương thức vẽ background:

|  |
| --- |
| void graphics::backGround(QPainter & painter){  int h =height();  int w = width();  painter.setBrush(QBrush("#8787af"));  painter.drawRect(0,0,w,h);  } |

* + Vẽ hình chữ nhật tại điểm (0,0) với chiều rộng là w, chiều dài là h và tô màu.
* Phương thức vẽ núi:

|  |
| --- |
| void graphics::mountain(QPainter &painter){  int h= height();  int w= width();  QString filename= "C:/xampp/htdocs/GitHub/DHMT/mountain.png";  painter.drawPixmap(10,175,w-20,h/2,QPixmap(filename));  } |

* + Vẽ núi tại vị trí (10,175) với chiều rộng là w-20, chiều dài là h/2 bằng hình ảnh có sẵn thông qua phương thức drawPixmap của lớp QPainter.
* Phương thức vẽ biển:

|  |
| --- |
| void graphics::sea(QPainter &painter){  int h= height();  int w= width();  painter.setBrush(QBrush("#8787ff"));  painter.drawRect(0,h-30,w,100);  } |

* + Vẽ hình chữ nhật tại vị trí (0, h-30) với chiều rộng là w, chiều cao là 100 và tô màu bằng phương thức setBrush với màu là #8787ff.
* Phương thức vẽ mặt trăng:

|  |
| --- |
| void graphics::moon(QPainter &painter , QPoint p){  QString filename = "C:/xampp/htdocs/GitHub/DHMT/moon.png";  painter.drawPixmap(p.x(),p.y(),30\*1.85,40,QPixmap(filename));  } |

* + Nhận vào vị trí vẽ là p, và vẽ mặt trăng tại vị trí p đó với kích thước là 30\*1.85 x 40 bằng phương thức vẽ drawPixmap.
* Phương thức vẽ chim:

|  |
| --- |
| void graphics::bird(QPainter& painter,QPoint p){  QString wingLeftOfFirstBird = "C:/xampp/htdocs/GitHub/DHMT/wing.png";  painter.drawPixmap(p.x(),p.y(),20\*(258/183),20,QPixmap(wingLeftOfFirstBird));  QString wingRightOfFirstBird = "C:/xampp/htdocs/GitHub/DHMT/wing.png";  painter.drawPixmap(p.x()+18\*(258/183),p.y(),20\*(258/183),20,QPixmap(wingRightOfFirstBird));  QPoint PSecond(p.x()+25,p.y()+25);  QString wingLeftOfSecondBird = "C:/xampp/htdocs/GitHub/DHMT/wing.png";  painter.drawPixmap(PSecond.x(),PSecond.y(),20\*(258/153),20,QPixmap(wingLeftOfSecondBird));  QString wingRightOfSecondBird = "C:/xampp/htdocs/GitHub/DHMT/wing.png";  painter.drawPixmap(PSecond.x()+18\*(253/183),PSecond.y(),20\*(258/153),20,QPixmap(wingRightOfSecondBird));  } |

* + Nhận vào vị trí p để vẽ chim.
  + Vẽ cánh bên trái của chim thứ nhất tại vị trí p với kích thước 20 \*(258/183) x 20 bằng hình ảnh có sẵn.
  + Vẽ cánh bên phải của chim thứ nhất vị trí p.x() cộng thêm chiều rộng của cánh trái.
  + Vẽ con chim thứ hai tương tự như trên.
* Phương thức vẽ mây:

|  |
| --- |
| void graphics::cloud(QPainter &painter, QPoint p, int size){  painter.setBrush(Qt::white);  painter.setPen(Qt::NoPen);  painter.drawEllipse(p.x(), p.y(), size, size);  painter.drawEllipse(p.x()+size/2, p.y(), size, size);  painter.drawEllipse(p.x()+size/2, p.y()+size/2, size, size);  painter.drawEllipse(p.x()+size, p.y()+size/2, size, size);  painter.drawEllipse(p.x()+size, p.y(), size, size);  painter.drawEllipse(p.x()+2/3\*size, p.y()+size/2, size, size);  painter.drawEllipse(p.x()+1.8\*size, p.y()+size/3, size, size);  painter.drawEllipse(p.x()-size/3, p.y()+size/3, size, size);  } |

* Nhận vào điểm vẽ p và kích thước size.
* Ý tưởng: vẽ các hình tròn chồng lên nhau và tô màu trắng.
* Phương thức vẽ thái cực:

|  |
| --- |
| void graphics::thaiCuc(QPainter &painter, QPoint p, int x, int y){  painter.setPen(Qt::NoPen);  painter.setBrush(QBrush(Qt::black));  painter.drawChord(p.x(),p.y(), x, y, 16\*0, 16\*180);  painter.setBrush(QBrush(Qt::white));  painter.drawChord(p.x(),p.y(), x, y, 16\*180, 16\*180);  QPoint a,b;  a.setX(p.x()+x/4);  a.setY(p.y()+y/2);  b.setX(p.x()+3\*x/4);  b.setY(p.y()+y/2);  painter.setBrush(QBrush(Qt::white));  painter.drawChord(p.x(), p.y()+y/4, x/2, y/2, 16\*0, 16\*180);  painter.setBrush(QBrush(Qt::black));  painter.drawChord(p.x()+x/2, p.y()+y/4, x/2, y/2, 16\*180, 16\*180);  painter.setBrush(QBrush(Qt::black));  painter.drawEllipse(a, x/20, x/20);  painter.setBrush(QBrush(Qt::white));  painter.drawEllipse(b, x/20, x/20);  } |

* Nhận vào điểm vẽ p, chiều dài và chiều rộng của thái cực.
* Ý tưởng:
  + Chia hình tròn lớn thành hai nửa vòng tròn nhỏ, nửa trên tô màu đen và nửa dưới tô màu trắng.
  + Sau đó vẽ thêm hai nửa cung tròn nữa nằm chồng lên hai cung tròn lớn.
  + Cuối cùng là vẽ hai hình tròn nhỏ.
  + Vẽ và tô màu cho các cung tròn bằng hàm drawChord.
* Phương thức vẽ sao:

|  |
| --- |
| void graphics::star(QPainter &painter, QPoint p, int size){  painter.setPen(Qt::NoPen);  painter.setBrush(QColor(random(255), random(255), random(255)));  QPolygon starWing1\_1;  QPoint a(p.x()-size\*sqrt(3)/2, p.y()-size/2);  QPoint b(p.x()-size\*sqrt(3)/2, p.y()+size/2);  QPoint c(p.x()-size\*sqrt(3)/2-2\*size, p.y());  starWing1\_1.push\_back(p);  starWing1\_1.push\_back(a);  starWing1\_1.push\_back(b);  painter.drawPolygon(starWing1\_1);  QPolygon starWing1\_2;  starWing1\_2.push\_back(a);  starWing1\_2.push\_back(b);  starWing1\_2.push\_back(c);  painter.drawPolygon(starWing1\_2);  for(int i=0; i<3; i++){  QPolygon points1, points2;  a=quay(a, p, 90);  b=quay(b, p, 90);  c=quay(c, p, 90);  points1.push\_back(p);  points1.push\_back(a);  points1.push\_back(b);  points2.push\_back(a);  points2.push\_back(b);  points2.push\_back(c);  painter.drawPolygon(points1);  painter.drawPolygon(points2);  }  } |

* Nhận vào vị trí p là tâm của ngôi sao và kích thước size.

a

p

c

b

* Vẽ một cánh sao như hình trên.
* Thực hiện vòng lặp for quay 4 điểm trên với góc 90 độ quanh p.
* Phương thức vẽ mặt trời:

|  |
| --- |
| void graphics::sun(QPainter &painter, QPoint p, int size){  if(p.y()>height()){  painter.setPen(Qt::NoPen);  painter.setBrush(Qt::NoBrush);  luminosity=0;  QPoint C(width()/2,0);  QPoint moonPoint(width()/2-200,0);  moonPoint=quay(moonPoint,C,moonAngle);  moon(painter,moonPoint);  }  else{  painter.drawEllipse(p, size+ luminosity, size+luminosity);  painter.drawEllipse(p, size+luminosity/2, size+luminosity/2);  painter.setPen(Qt::NoPen);  painter.setBrush(Qt::yellow);  painter.drawEllipse(p, size-20, size-20);  }  } |

* Nhận vào điểm p là vị trí vẽ mặt trời, kích thước size.
* Ý tưởng:
* Sử dụng hàm drawEllipse để vẽ mặt trời và độ lan tỏa ánh sáng của mặt trời.
* Khi mặt trời lặn (p.y() >height()) thì ẩn mặt trời, vẽ mặt trăng với tâm C(width()/2,0) và góc quay moonAngle được khai báo ở timerEvent.
* Phương thức vẽ thuyền:

|  |
| --- |
| void graphics::boat(QPainter &painter, QPoint p, int size){  QPolygon points;  points.push\_back(p);  QPoint a(p.x()+size, p.y());  QPoint b(p.x()+size\*4/5, p.y()+size/7);  QPoint c(p.x()+size\*1/5, p.y()+size/7);  points.push\_back(a);  points.push\_back(b);  points.push\_back(c);  painter.setBrush(QBrush("#715842"));  painter.drawPolygon(points);  //Cot buom  QPoint dinhCotBuom=trungdiem(p,a);  dinhCotBuom.setY(dinhCotBuom.y()-size\*3/5);  painter.setBrush(QBrush("#353847"));  painter.drawRect(dinhCotBuom.x(), dinhCotBuom.y(), size/30, size/1.75);  //Canh buom  QPolygon canhBuom1;  canhBuom1.push\_back(QPoint(dinhCotBuom.x()-size/70, dinhCotBuom.y()+size/23));  canhBuom1.push\_back(QPoint(dinhCotBuom.x()-size/70, dinhCotBuom.y()+size\*2/5));  canhBuom1.push\_back(QPoint(dinhCotBuom.x()-size/5, dinhCotBuom.y()+size\*2/5));  painter.setBrush(QBrush("#00875f"));  painter.drawPolygon(canhBuom1);  QPolygon canhBuom2;  canhBuom2.push\_back(QPoint(dinhCotBuom.x()+size/23, dinhCotBuom.y()+15));  canhBuom2.push\_back(QPoint(dinhCotBuom.x()+size/23, dinhCotBuom.y()+size\*0.5));  canhBuom2.push\_back(QPoint(dinhCotBuom.x()+size/4, dinhCotBuom.y()+size\*0.5));  painter.drawPolygon(canhBuom2);  //Co  painter.setBrush(QBrush(Qt::red));  painter.drawRect(dinhCotBuom.x()+size/25, dinhCotBuom.y(), size/7, size/9);  QPoint hinhCo(dinhCotBuom.x()+size/25+size/27, dinhCotBuom.y()+size/27);  thaiCuc(painter, hinhCo, size/17, size/17);  } |

* Nhận vào điểm p để vẽ vị trí của thuyền, kích thước size.
* Ý tưởng:
  + Dùng hàm drawPolygon để vẽ thân thuyền. Với điểm p(x,y), a(x+4/5\*size, y+size/7), b(x+size\*4/5, y+size/7), c(x+size\*1/5, y+size/7).
  + Xác định vị trí để vẽ cột buồm là trung điểm giữa p và a. Sử dụng hàm drawRect để vẽ.
  + Từ đỉnh cột buồm vẽ hai cánh buồm ở hai bên cột buồm.
  + Từ đỉnh cột buồm vẽ lá cờ.

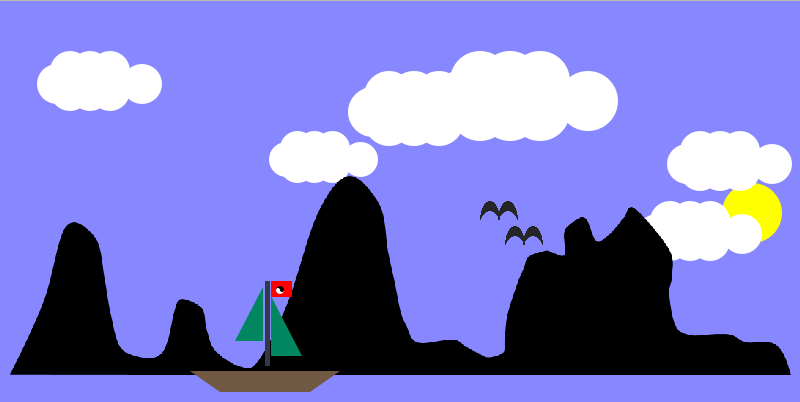
a

p

b

c

## Kết quả.



Hình 1: Khung cảnh vào ban ngày



Hình 2: Khung cảnh vào ban đê

# CHƯƠNG 4 : KẾT QUẢ ĐẠT ĐƯỢC

## Kết quả

* Đã vẽ được cảnh núi, thuyền, biển cũng hiệu ứng di chuyển của mặt trời và mặt trăng.
* Vẽ được bầu trời đầy sao lấp lánh vào ban đêm.

## Hạn chế

* Các đối tượng vẽ chưa được phức tạp so với hình ảnh thực tế của các đối tượng đó.
* Chưa có giao diện thân thiện để có thể thay đổi tốc độ di chuyển hay kích thước của các đổi tượng.

## Hướng phát triển

* Cải thiện tính phức tạp cho các đối tượng.
* Thêm giao diện để thay đổi kích thước hay tốc độ di chuyển của các đối tượng.

# TÀI LIỆU THAM KHẢO

1. Phạm Xuân Hiền. (n.d.). *Giáo trình Đồ Họa Máy Tính.*
2. *Qt Creator Manual*. (n.d.). Retrieved from http://doc.qt.io/qtcreator/

# PHỤ LỤC

* *Graphics.h*

|  |
| --- |
| #ifndef GRAPHICS\_H  #define GRAPHICS\_H  #include <QWidget>  class graphics : public QWidget  {  Q\_OBJECT  public:  explicit graphics(QWidget \*parent = 0);  void *paintEvent*(QPaintEvent \*);  void *timerEvent*(QTimerEvent \*);  void mountain(QPainter& painter);  void sea(QPainter& painter);  void moon(QPainter& painter,QPoint p);  void bird(QPainter& painter, QPoint p);  QPoint quay(QPoint p,QPoint c,int deta);  QPoint trungdiem(QPoint a, QPoint b);  int random(int n);  int randomNM(int n,int m);  void backGround(QPainter & painter);  void thaiCuc(QPainter &painter, QPoint p, int x, int y);  void sun(QPainter &painter, QPoint p, int size);  void boat(QPainter &painter, QPoint p, int size);  void cloud(QPainter &painter, QPoint p, int size);  void star(QPainter &painter, QPoint p, int size);  int speedOfBoat;  int speedOfBird;  int speedOfCloud1;  int speedOfCloud2;  int luminosity;  int sunAngle;  double timerId;  int moonAngle;  signals:  public slots:  };  #endif // GRAPHICS\_H |

* *Graphics.cpp*

|  |
| --- |
| #include "graphics.h"  #include <QPainter>  #include <QPoint>  #include <math.h>  #include <QTableWidgetItem>  #include <QPen>  #include <QString>  #include <QLineEdit>  #include <QStyleOption>  graphics::graphics(QWidget \*parent) :  QWidget(parent)  {  timerId=startTimer(250);  luminosity=0;  sunAngle =0;  moonAngle =180;  speedOfBoat=0;  speedOfBird=0;  }  void graphics::*timerEvent*(QTimerEvent \*){  speedOfBoat+=5;  speedOfBird +=10;  speedOfCloud1 +=10;  speedOfCloud2 +=8;  sunAngle +=4;  moonAngle-=4;  if(luminosity==30\*45){  luminosity = 0;  }  else{  luminosity +=30;  }  repaint();  }  void graphics::*paintEvent*(QPaintEvent \*){  QPainter painter(this);  painter.setRenderHint(QPainter::Antialiasing);  backGround(painter);  for(int i=1;i<=1000;i++){  int w =370;  QPoint p(i+1,random(w));  star(painter,p,randomNM(1,2));  i+=10;  }  painter.setBrush(QBrush("#8787ff"));  QPoint sunP(0,height());  QPoint O(width()/2,height());  sunP = quay(sunP,O,sunAngle);  sun(painter,sunP,50);  sea(painter);  QPoint cloud1(300,130);  cloud1.setX(cloud1.x()+ speedOfCloud1);  QPoint cloud2(650,200);  QPoint cloud3(450,50);  QPoint cloud4(280,130);  QPoint cloud5(50,50);  QPoint cloud6(60,70);  cloud6.setX(cloud6.x()+ speedOfCloud2);  cloud(painter,cloud1,40);  cloud(painter,cloud2,40);  cloud(painter,cloud3,60);  cloud(painter,cloud4,35);  cloud(painter,cloud5,40);  cloud(painter,cloud6,50);  mountain(painter);  QPoint boatP(0,370);  boatP.setX(boatP.x()+ speedOfBoat);  boat(painter,boatP,150);  QPoint birdP(100,200);  birdP.setX(birdP.x()+ speedOfBird);  bird(painter,birdP);  }  QPoint graphics::quay(QPoint p,QPoint c,int deta){  QPoint pnew;  double goc=deta\*3.14/180;  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;  }  QPoint graphics::trungdiem(QPoint a, QPoint b){  QPoint pnew((a.x()+b.x())/2, (a.y()+b.y())/2);  return pnew;  }  void graphics::mountain(QPainter &painter){  int h= height();  int w= width();  QString filename= "../Img/mountain.png";  painter.drawPixmap(10,175,w-20,h/2,QPixmap(filename));  }  void graphics::moon(QPainter &painter , QPoint p){  QString filename = "../Img/moon.png";  painter.drawPixmap(p.x(),p.y(),30\*1.85,40,QPixmap(filename));  }  void graphics::sea(QPainter &painter){  int h= height();  int w= width();  painter.setBrush(QBrush("#8787ff"));  painter.drawRect(0,h-30,w,100);  }  void graphics::bird(QPainter& painter,QPoint p){  QString wingLeftOfFirstBird = "../Imgwing.png";  painter.drawPixmap(p.x(),p.y(),20\*(258/183),20,QPixmap(wingLeftOfFirstBird));  QString wingRightOfFirstBird = "../Img/wing.png";  painter.drawPixmap(p.x()+18\*(258/183),p.y(),20\*(258/183),20,QPixmap(wingRightOfFirstBird));  QPoint PSecond(p.x()+25,p.y()+25);  QString wingLeftOfSecondBird = "../Img/wing.png";  painter.drawPixmap(PSecond.x(),PSecond.y(),20\*(258/153),20,QPixmap(wingLeftOfSecondBird));  QString wingRightOfSecondBird = "../Img/wing.png";  painter.drawPixmap(PSecond.x()+18\*(253/183),PSecond.y(),20\*(258/153),20,QPixmap(wingRightOfSecondBird));  }  void graphics::backGround(QPainter & painter){  int h =height();  int w = width();  painter.setBrush(QBrush("#8787af"));  painter.drawRect(0,0,w,h);  }  //Random số  int graphics::random(int n)  {  return rand() % n;  }  int graphics::randomNM(int n,int m){  return rand()% m+n;  }  void graphics::cloud(QPainter &painter, QPoint p, int size){  painter.setBrush(Qt::white);  painter.setPen(Qt::NoPen);  painter.drawEllipse(p.x(), p.y(), size, size);  painter.drawEllipse(p.x()+size/2, p.y(), size, size);  painter.drawEllipse(p.x()+size/2, p.y()+size/2, size, size);  painter.drawEllipse(p.x()+size, p.y()+size/2, size, size);  painter.drawEllipse(p.x()+size, p.y(), size, size);  painter.drawEllipse(p.x()+2/3\*size, p.y()+size/2, size, size);  painter.drawEllipse(p.x()+1.8\*size, p.y()+size/3, size, size);  painter.drawEllipse(p.x()-size/3, p.y()+size/3, size, size);  }  void graphics::star(QPainter &painter, QPoint p, int size){  painter.setPen(Qt::NoPen);  painter.setBrush(QColor(random(255), random(255), random(255)));  QPolygon starWing1\_1;  QPoint a(p.x()-size\*sqrt(3)/2, p.y()-size/2);  QPoint b(p.x()-size\*sqrt(3)/2, p.y()+size/2);  QPoint c(p.x()-size\*sqrt(3)/2-2\*size, p.y());  starWing1\_1.push\_back(p);  starWing1\_1.push\_back(a);  starWing1\_1.push\_back(b);  painter.drawPolygon(starWing1\_1);  QPolygon starWing1\_2;  starWing1\_2.push\_back(a);  starWing1\_2.push\_back(b);  starWing1\_2.push\_back(c);  painter.drawPolygon(starWing1\_2);  for(int i=0; i<3; i++){  QPolygon points1, points2;  a=quay(a, p, 90);  b=quay(b, p, 90);  c=quay(c, p, 90);  points1.push\_back(p);  points1.push\_back(a);  points1.push\_back(b);  points2.push\_back(a);  points2.push\_back(b);  points2.push\_back(c);  painter.drawPolygon(points1);  painter.drawPolygon(points2);  }  }  void graphics::boat(QPainter &painter, QPoint p, int size){  QPolygon points;  points.push\_back(p);  QPoint a(p.x()+size, p.y());  QPoint b(p.x()+size\*4/5, p.y()+size/7);  QPoint c(p.x()+size\*1/5, p.y()+size/7);  points.push\_back(a);  points.push\_back(b);  points.push\_back(c);  painter.setBrush(QBrush("#715842"));  painter.drawPolygon(points);  //Cot buom  QPoint dinhCotBuom=trungdiem(p,a);  dinhCotBuom.setY(dinhCotBuom.y()-size\*3/5);  painter.setBrush(QBrush("#353847"));  painter.drawRect(dinhCotBuom.x(), dinhCotBuom.y(), size/30, size/1.75);  //Canh buom  QPolygon canhBuom1;  canhBuom1.push\_back(QPoint(dinhCotBuom.x()-size/70, dinhCotBuom.y()+size/23));  canhBuom1.push\_back(QPoint(dinhCotBuom.x()-size/70, dinhCotBuom.y()+size\*2/5));  canhBuom1.push\_back(QPoint(dinhCotBuom.x()-size/5, dinhCotBuom.y()+size\*2/5));  painter.setBrush(QBrush("#00875f"));  painter.drawPolygon(canhBuom1);  QPolygon canhBuom2;  canhBuom2.push\_back(QPoint(dinhCotBuom.x()+size/23, dinhCotBuom.y()+15));  canhBuom2.push\_back(QPoint(dinhCotBuom.x()+size/23, dinhCotBuom.y()+size\*0.5));  canhBuom2.push\_back(QPoint(dinhCotBuom.x()+size/4, dinhCotBuom.y()+size\*0.5));  painter.drawPolygon(canhBuom2);  //Co  painter.setBrush(QBrush(Qt::red));  painter.drawRect(dinhCotBuom.x()+size/25, dinhCotBuom.y(), size/7, size/9);  QPoint hinhCo(dinhCotBuom.x()+size/25+size/27, dinhCotBuom.y()+size/27);  thaiCuc(painter, hinhCo, size/17, size/17);  }  void graphics::sun(QPainter &painter, QPoint p, int size){  if(p.y()>height()){  painter.setPen(Qt::NoPen);  painter.setBrush(Qt::NoBrush);  luminosity=0;  QPoint C(width()/2,0);  QPoint moonPoint(width()/2-200,0);  moonPoint=quay(moonPoint,C,moonAngle);  moon(painter,moonPoint);  }  else{  painter.drawEllipse(p, size+ luminosity, size+luminosity);  painter.drawEllipse(p, size+luminosity/2, size+luminosity/2);  painter.setPen(Qt::NoPen);  painter.setBrush(Qt::yellow);  painter.drawEllipse(p, size-20, size-20);  }  }  void graphics::thaiCuc(QPainter &painter, QPoint p, int x, int y){  painter.setPen(Qt::NoPen);  painter.setBrush(QBrush(Qt::black));  painter.drawChord(p.x(),p.y(), x, y, 16\*0, 16\*180);  painter.setBrush(QBrush(Qt::white));  painter.drawChord(p.x(),p.y(), x, y, 16\*180, 16\*180);  QPoint a,b;  a.setX(p.x()+x/4);  a.setY(p.y()+y/2);  b.setX(p.x()+3\*x/4);  b.setY(p.y()+y/2);  painter.setBrush(QBrush(Qt::white));  painter.drawChord(p.x(), p.y()+y/4, x/2, y/2, 16\*0, 16\*180);  painter.setBrush(QBrush(Qt::black));  painter.drawChord(p.x()+x/2, p.y()+y/4, x/2, y/2, 16\*180, 16\*180);  painter.setBrush(QBrush(Qt::black));  painter.drawEllipse(a, x/20, x/20);  painter.setBrush(QBrush(Qt::white));  painter.drawEllipse(b, x/20, x/20);  } |

* *Mainwindow.h*

|  |
| --- |
| #ifndef MAINWINDOW\_H  #define MAINWINDOW\_H  #include <QMainWindow>  namespace Ui {  class MainWindow;  }  class MainWindow : public QMainWindow  {  Q\_OBJECT  public:  explicit MainWindow(QWidget \*parent = 0);  ~*MainWindow*();  private:  Ui::MainWindow \*ui;  };  #endif // MAINWINDOW\_H |

* *Main.cpp*

|  |
| --- |
| #include "mainwindow.h"  #include <QApplication>  int main(int argc, char \*argv[])  {  QApplication a(argc, argv);  MainWindow w;  w.show();  return a.exec();  } |

* *Mainwindow.cpp*

|  |
| --- |
| #include "mainwindow.h"  #include "ui\_mainwindow.h"  MainWindow::MainWindow(QWidget \*parent) :  QMainWindow(parent),  ui(new Ui::MainWindow)  {  ui->setupUi(this);  }  MainWindow::~*MainWindow*()  {  delete ui;  } |