O’ZBEKISTON RESPUBLIKASI

OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

_____ DAVLAT UNIVERSITETI

“AMALIY MATEMATIKA VA INFORMATIKA” KAFEDRASI

“5130200 – Amaliy matematika va informatika” ta’lim yo’nalishi bo’yicha

bakalavr darajasini olish uchun

_____*ning*

«* C++ tilining multimedia imkoniyatlaridan foydalanish*»

mavzusida yozgan

BITIRUV MALAKAVIY ISHI

Ilmiy rahbar: _____

“*Himoyaga tavsiya etildi*”

Fizika-matematika fakulteti dekani

_____ prof. _____

“___” _____ 20__-y.

TOSHKENT – 20__-yil

MUNDARIJA

KIRISH

Mavzuning dolzarbligi: Zamonaviy dasturlash tillarining asosini tashkil etuvchi obyektga yo'naltirilgan dasturlash (OYD)- hozirgi kundagi dasturlashga bo'lgan yangicha yondashuvdir. Hisoblash texnikasining rivojlanishi va yechilayotgan masalalarni tobora murakkablashuvi dasturlashning turli modellarini (paradigmalarini) yuzaga kelishiga sabab bo'lmoqda. Birinchi kompilyatorlar (masalan, FORTRAN tili) dasturlashning funksiyalardan foydalanishga asoslangan protsedura modelini qo'llab quvvatlagan. Bu model yordamida dastur tuzuvchi bir necha ming qatorli dasturlarni yozishi mumkin edi. Rivojlanishning keyingi bosqichida dasturlarning strukturali modeli paydo bo'ldi va ular ALGOL, Pascal va C tillar kompilyatorlarida o'z aksini topdi. Strukturali dasturlashning mohiyati – dasturni o'zaro bog'langan protseduralar (bloklar) va ular qayta ishlaydigan berilganlarning majmuasi deb qarashidir. Ushbu model dastur bloklarini keng qo'llashga, GOTO operatoridan imkon qadar kam foydalanishga tayangan va unda dastur tuzuvchi o'n ming qatordan ortiq buyruqlardan iborat dasturni yarata oladi. Yaratilgan dasturni protsedurali modelga nisbatan sozlash va nazorat qilish oson kechadi. Bularga qaramasdan yangi murakkab masalalarning paydo bo'lishi dasturlashdagi yangi yondashuvning paydo bo'lishiga olib keldi. Ya'ni murakkab masalalarni yechish uchun dasturlashning yangi uslubiga zarurat paydo bo'lib, u OYD modelida amalga oshirildi. Ushbu ishda esa OYD ning imkoniyatlaridan foydalanib grafika va multimedia bilan ishlash jarayonini qarab chiqamiz. Bugungi kunga kelib grafika va multimedia bilan ishlovchi ko'plab dasturiy vositalar mavjud bo'lsada, kompyuter bilan bog'liqlikda ishlovchi qurilmalarning mukammallashuvi grafika va multimedia bilan ishlovchi yangi dasturiy vositalarga ehtiyoj sezmoqda. Bunga misol qilib tibbiyot, kuzatuv obyektlari kabilarni aytish mumkin. Shu boisdan ham ishda qo'yilgan masala ahamiyatlidir.

Ishning maqsadi: C++ dasturlash tilining grafika va multimedia imkoniyatlarini o'rganish hamda ularni amaliy dasturlar qurishga tatbiq etish.

Ishda qo’yilgan vazifalar:

− Tilning grafika va multimedia bilan ishlovchi komponentalarini o’rganish;

− Bitli tasvirlar va multiplikatsiyalar bilan ishlashni batafsil o’rganish;

−amaliy dasturlarda grafika, animatsiya, ovozli va video fayllarni boshqarish jarayonlarini tashkil etish;

Tadqiqotning ilmiy yangiliklari :

Mukammal dasturiy vositalar qurishda C++ tilining grafika va multimedia imkoniyatlaridan foydalanish.

Tadqiqotning predmeti va ob’ekti: Uzluksiv ta’lim tizimining barcha bo’g’inlariga tegishli bo’lgan multimediali elektron o’quv adabiyotlar, animatsiya, ovoz va video fayllar bilan ishlovchi mukammal dasturlar.

Tadqiqotning ilmiy ahamiyati:

−mavzu bo’yicha ilmiy-uslubiy, nazariy adabiyotlarni o’rganish;

− ta’lim to’g’risida davlat hujjatlari, DTS talablari, ilg’or mutaxassis olimlarning fikrlarini o’rganish;

− mavzuga aloqador mavjud internet resurslaridan foydalanish;

− C++ tilining grafika va multimedia imkoniyatlarini amaliy dasturlar qurishga tatbiq etish.

Ishning hajmi va strukturasi: Bitiruv malakaviy ishi, kirish, uchta bob, xulosa, foydalanilgan adabiyotlar hamda internet resurslari ro’yxatidan iborat. Ishning 1-bobida grafik asos va primitivlar, tasvirni tashqi fayldan o’qish, bitli tasvirlardan foydalanish haqida umumiy tushunchalar keltiriladi. 2-bobda multiplikatsiya, multimediali fayllar bilan ishlashda Animate komponenetasining qo’llanilishi, multimediali fayllar bilan ishlashda MediaPlayer komponentasining qo’llanilishi

amaliy masalalar yordamida tushuntiriladi. 3-bobida bir nechta funksiyalarning grafiklarini bir vaqtda tekislikda tasvirlash, ovozli fayllar bilan ishlash, video fayllar bilan ishlash dasturining matni va undan foydalanib olingan natijalardan namunalar keltiriladi.

I BOB C++ BUILDER DASTURINING GRAFIK IMKONIYATLARI

C++ Builder muhiti dasturchiga grafik imkoniyatlarni ham taqdim etadi . Ushbu Bitiruv malakaviy ishda C++ tilining multimedia imkoniyatlarini yoritish maqsadida forma sirtida garafik chizish, uning ustida rasm yoki fotosuratni joylashtirish, grafik yoki rasm ko'rinishidagi obyektlar nusxasini skanerlash kabi masalalarni qaraymiz.

1.1. Grafik asos va primitivlar

Asos. Dasturlash tili *Canvas* xususiyatlariga mos keluvchi grafikani forma sirtida chizadi. Shuning uchun *Canvas* chizishning asosi hisoblanadi. *Canvas* ning mos metodlaridan foydalaninb, forma sirtida aylana, to'g'ri chiziq, to'rtburchak yoki boshqa primitivlarni chizish mumkin[1,2]. Masalan:

Form1->Canvas->Rectangle(10,10,50,50);

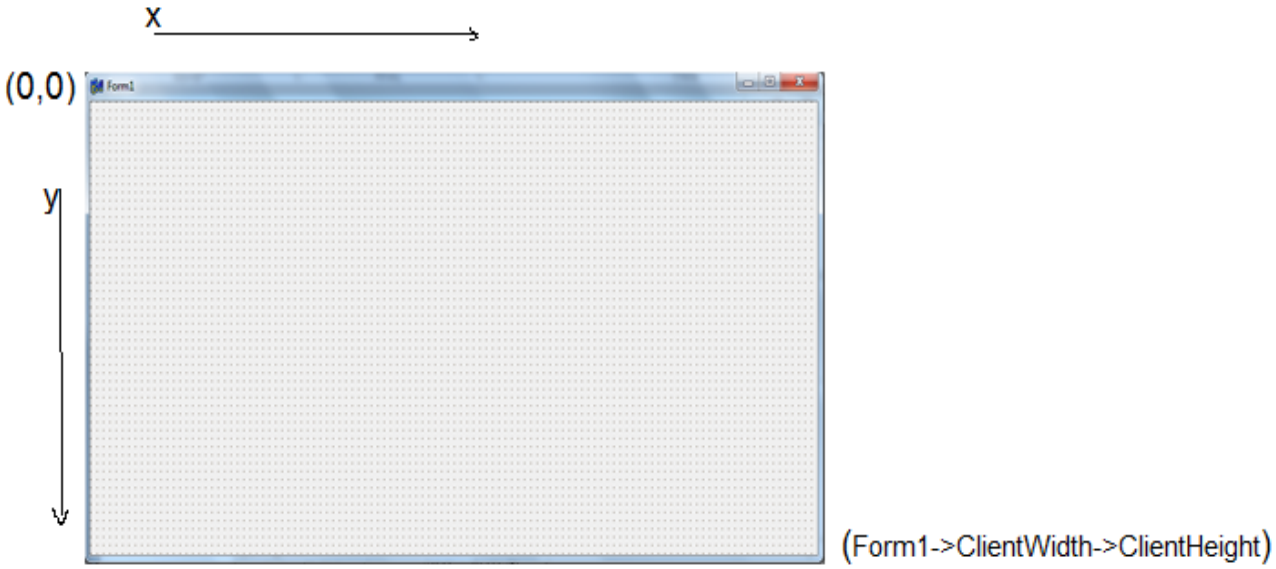
Operatori forma sirtida to'g'ri to'rtburchak chizadi.

1—Jadval. Grafik primitivlarning ishlatish metodlari

Metod	**Vazifasi**
LineTo(x,y);	Joriy nuqtadan ko'rsatilgan nuqtagacha bo'lgan masofada to'g'ri chiziq chizadi
Rectangle(x1,y1,x2,y2);	Yuqori chap burchagi x_1,y_1 quyi o'ng burchagi x_2,y_2 bo'lgan to'g'ri to'rtburchak chizadi
FillRect(x1,y1,x2,y2);	Yuqori chap burchagi x_1,y_1 quyi o'ng burchagi x_2,y_2 bo'lgan ichi bo'yalgan to'g'ri to'rtburchak chizadi
FrameRect(x1,y1,x2,y2);	To'g'ri to'rtburchak konturini chizadi

RoundRect(x1,y1,x2,y2,x3,y3);	Burchaklari silliq bo'lgan to'g'ri to'rtburchak chizadi
Ellipse(x1,y1,x2,y2)	Ellips yoki aylana chizadi.x_1,x_2,y_1,y_2-lar ellipsga chizilayotgan ellipsga tashqi chizilgan to'g'ri to'rtburchak(kvadrat) ning diagonal uchlari koordinatalri
Polyline(points,n)	Siniq chiziq chizish. Bu yerda points TPoint turidagi massiv. Massivning har bir elementidagi yozuv o'zida x_i va y_i sinish nuqtalarining koordinatalarini saqlaydi. n esa siniq chiziqlar soni

Boshqa grafik primitivlarni chizish uchun *Canvas* metodining ro'yxatiga qarash kerak bo'ladi. Asos alohida nuqtalar(piksellar)dan tuzilgan bo'ladi.Asos tekisligidagi piksel koordinatalari *X*—gorizontal va *Y*—vertikal o'qlar bo'yicha aniqlanadi. Koordinatalar yuqori chap burchakdan boshlanib, chapdan o'ngga va yuqoridan pastga o'sib boradi (1-rasm).Ya'ni yuqori chap burchak *(0,0)* koordinata ,quyi o'ng burchak esa *(Client Width,Client Heigth)* koordinatalar bilan aniqlanadi. Alohida nuqta (piksel) joylashuvi *Pixels* bilan aniqlanib uning x_i va y_i joylashuv elementlari massivi va rang parametrlaridan iborat xossasi mavjud. Grafik chizishni boshlashda *OnPaint* hodisasidan foydalanish mumkin. Buning uchun *"Object Inspector"* ning *"Events"* hodisalar bo'limidan *OnPaint* hodisasi ishga tushiriladi.



1.1. -rasm. Forma sirtining nuqta koordinatalari

Qalam va mo'yqalam. Grafik primitivlarni chizish metodlari faqat chizish jarayonini ta'minlaydi. Chizish metodidagi *Pen* va *Brush* xususiyatlari grafik element ko'rinishini aniqlaydi. Qalam va mo'yqalam mos ravishda *Canvas* metodining *Pen* va *Brush* obyektlari orqali aniqlanadi. *Pen* obyekti geometrik tasvirlovchi chiziqning yo'g'onligi va rangi kabi xossalarni o'zida saqlaydi.

2-Jadval. Pen (qalam) obyektining xususiyatlari

Xususiyatlar	**Vazifasi**
Color	Chiziq rangi
Width	Chiziq qalinligi
Style	Chiziq ko'rinishi (psSolid—yaxlit, psDasht—uzuq chiziq, psDot—qisqa-uzuq,psClear—ko'rinmas chiziq)

Brush obyekti to'rtburchak, aylana kabi yopiq sohalar ichini bo'yash uchun xizmat qiladi.

3-Jadval. Brush (mo'yqalam) obyektining xususiyatlari.

Xususiyatlar	**Vazifasi**
Color	Yopiq soha rangi
Style	Sohani bo'yash stili, bsSolid—yaxlit bo'yash, bsVertical—vertikal chiziq bilan qoplash, bsHorizintal—gorizontal chiziq bilan qoplash,bsFDiagonal—diagonal bo'yicha oldinga og'ishgan rangli chiziq bilan qoplash, bsBDiagonal — diagonal bo'yicha orqaga og'ishgan , bsCross—kesishgan chiziqlar bilan, bsDiagCross—diagonal kesishgan chiziqlar bilan bo'yash

Grafik primitivlar. Ekranda hosil qilinadigan ixtiyoriy grafik, rasmlar asosida primitiv shakllar yotadi. Bunday shakllarga nuqta, to'g'ri chiziq aylana, to'g'ri to'rtburchak kabilar kiradi. Quyida ularning ayrimlarini hosil qilishni ko'rib o'tamiz.

To'g'ri chiziqni chizish uchun *LineTo* metodidan foydalaniladi.Bu metod qalamning joriy koordinatasidan metodning argumentida ko'rsatilgan nuqtagacha bo'lgan chiziqni chizadi. Masalan:

Canvas->LineTo(100,200);

Qalamning joriy koordinatasini *MoveTo* metodi bilan o'zagartirish mumkin:

Misol:

Canvas->MoveTo(10,10);

Canvas->LineTo(50,10);

Bu dastur qismi *(10,10)* nuqtadan *(50,10)* nuqtagacha bo'lgan masofada kesma chizadi.

Polyline metodi yordamida siniq chiziq chizish mumkin. Bu metodning parametri sifatida siniq chiziq tugunlari koordinatalarini saqlovchi *TPoint* turidagi massiv hamda siniq chiziqlar sonini ko'rsatuvchi *n* beriladi. Masalan, quyidagi dastur matni 3 ta siniq chiziqni chizadi:

```
TPoint p[4];

p[0].x=100; p[0].y=100;   // boshlanishi

p[1].x=100; p[1].y=150;   //nuqta peregiba

p[2].x=150; p[2].y=150;      //nuqta peregiba

p[3].x=150; p[3].y=100;   // tugashi

Canvas->Polyline(p,3);
```

Polyline metodi yordamida siniq chiziqli yopiq kontur ham chizish mumkin. Buning uchun parametrdagi massivning birinchi va oxirgi elementlariga bir xil koordinatalar joylashtirish kerak.

Rectangle metodi to'g'ri to'rtburchak chizadi. Metod argumentida ko'rsatilgan parametrlar to'g'ri to'rtburchakning yuqori chap va quyi o'ng burchaklarining koordinatalari hisoblanadi. Misol:

```
Canvas->Rectangle(10,10,50,50);
```

Yuqorida ta'kidlab o'tganimizdek, to'g'ri to'rtburchakning chegara chiziqlaridagi ko'rinishlarini *Pen* yordamida, uning ichini bo'yashni *Brush* yordamida amalga oshirish mumkin. To'g'ri to'rtburchak chizishning boshqa bir necha metodlari ham mavjud bo'li, ular quyidagilar: *FillRect*, *FrameRect* va *RoundRect*.

Polygon metodi yordamida ko'pburchak chizish mumkin. Bu metodning umumiy ko'rinishi quyidagicha:

Canvas->Polygon(p,n);

Bu yerda *p—Tpoint* turidagi massiv bo'lib, kopburchak uchlarining koordinatalarini saqlaydi, *n*—uchlar soni

Quyida *Polygon* metodidan foydalanib , romb chizish matni keltirilgan:

TPoint p[4];

p[0].x=50;p[0].y=100;

p[1].x=150;p[1].y=75;

p[2].x=250;p[2].y=100;

p[3].x=150;p[3].y=125;

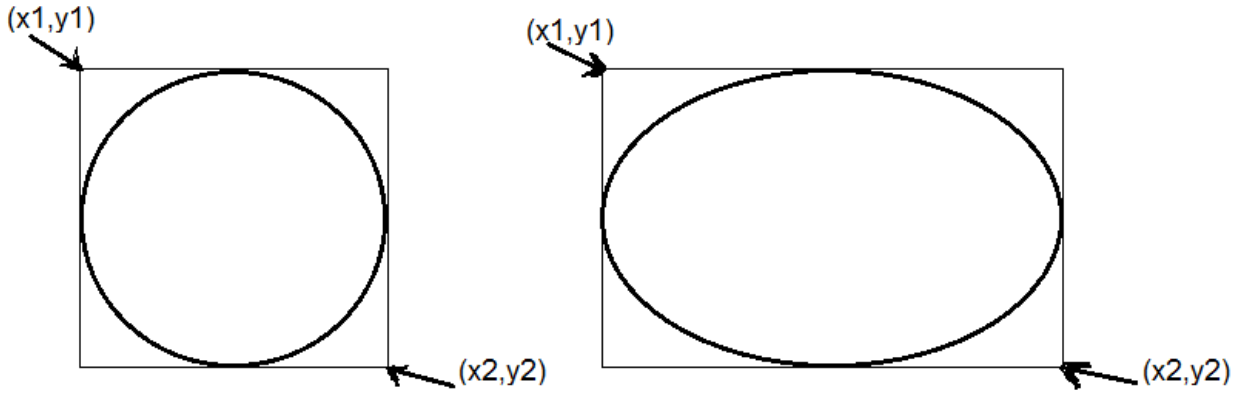
Canvas->Brush->Color=clRed;

Canvas->Polygon(p,3);

Aylana va ellips chizish uchun *Ellipse* metodidan foydalaniladi. Uning ko'rinishi quyidagicha:

Canvas->Ellipse(x1,y1,x2,y2);

Bu yerda *x1,y1,x2,y2* lar ellipsga tashqi chizilgan kvadratning koordinatalarini, yoki agar tashqi chizilgan to'g'ri to'rtburchakning koordinatalarini ifodalaydi.



1.2. Rasm. Aylana yoki ellipsning ko'rinishini aniqlaydigan *Ellipse* metodi parametrlarining qiymatlari

Bu 4 ta koordinata o'rnida bitta *TRect* tipli obyekt berish ham mumkin. Masalan:

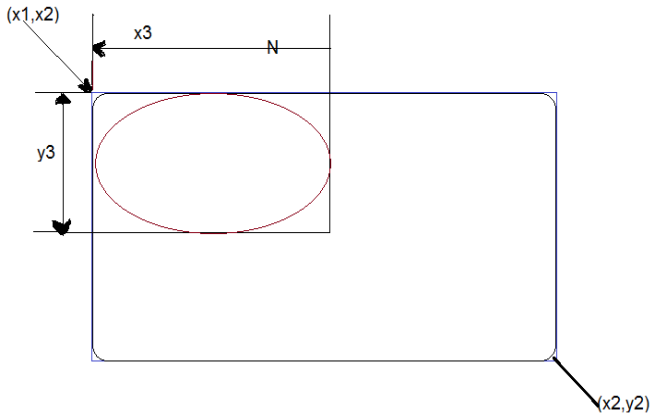
```
TRect rec=Rect(10,10,50,50);
Canvas->Ellipse(rec);
```

Boshqa primitivlarda qo'llanilgani kabi bu shaklda ham *Pen* va *Brush* metodlarini qo'llash mumkin.

Arc metodi aylana yoyini chizadi. Metodning umumiy ko'rinishi quyidagicha:

```
Canvas->Arc(x1,y1,x2,y2,x3,y3,x4,y4);
```

Bu yerda *x1,y1,x2,y2* parametrlar yoy ajratib olinayotgan ellipsni aniqlaydi, *x3,y3* yoyning boshlang'ich , *x4,y4* esa oxirgi nuqtalarining koordinatalari. *Arc* metodi yoy chizishni soat strelkasi yo'nalishida amalga oshiradi.

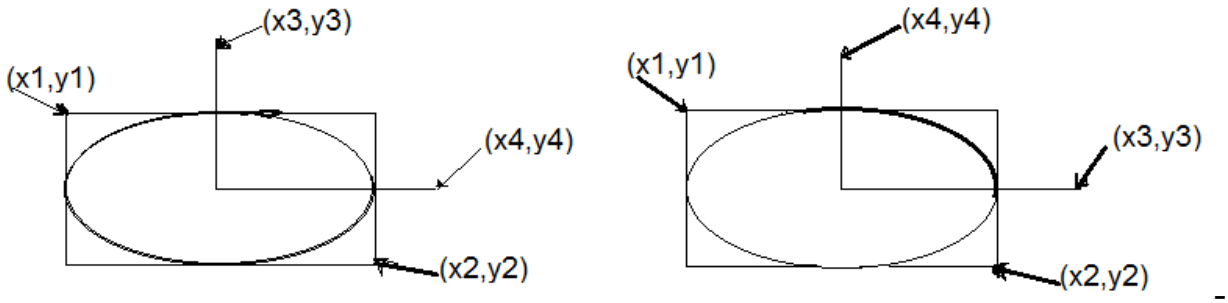


1.3.-rasm. *RoundRect* metodi yordamida geometrik figura ko'rinishini aniqlash

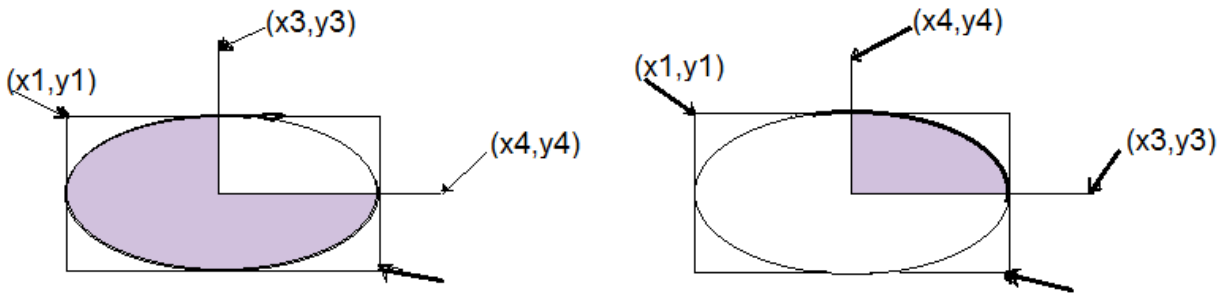
Pie metodi ellips yoki aylana sektorini chizadi. Bu metodning umumiy ko'rinishi quyidagicha :

Canvas->Pie(x1,y1,x2,y2,x3,y3,x4,y4);

Bu yerda *x1,y1,x2,y2*—sektor ajrati olinayotgan ellipsning koordinatalari, *x3,y3,x4,y4*—sektorning chegaralarining koordinatalari.



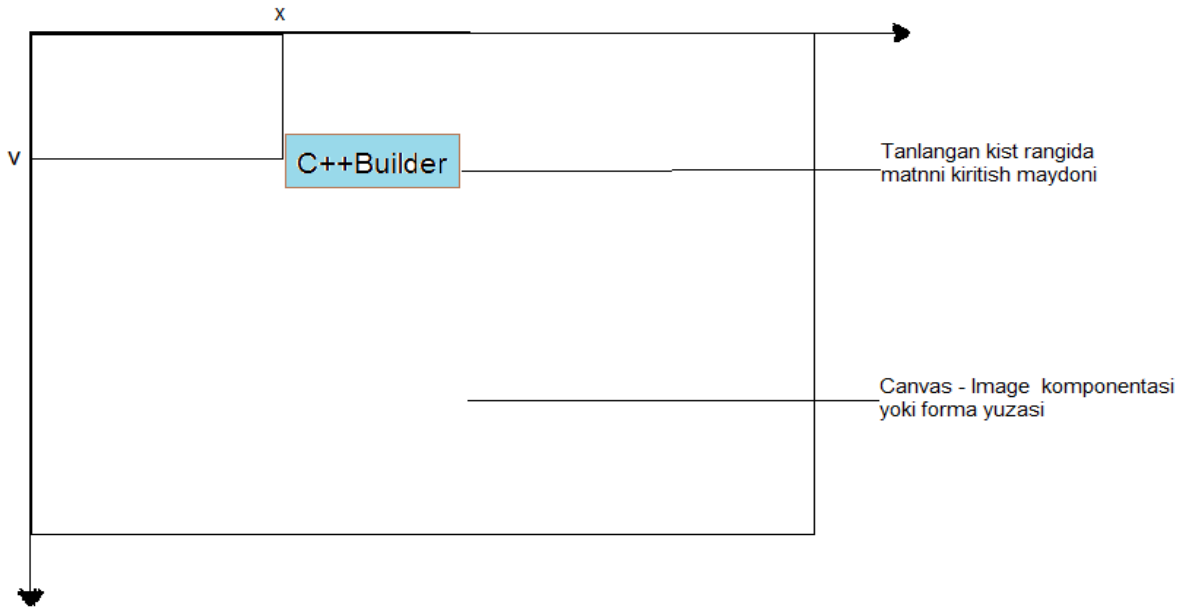
1.4. rasm. *Arc* metodi yordamida geometrik figura ko'rinishini aniqlash



1.5.-rasm *Pie* metodi yordamida geometrik figura ko'rinishini aniqlash

Grafik asos sirtiga matnli ma'lumotlarni chiqarish uchun *TextOutA* metodidan foydalaniladi. Bu metodning umumiy ko'rinishi quyidagicha:

TextOutA(x,y,"Matn");



Bu yerda *Matn* parametrida chiqariladigan matn joylashtiriladi. *X* va *Y* parametrlar chiqariladigan matnning boshlang'ich koordinatasi hisoblanadi. Matn shrift *Canvas* obyeektining *Font* metodi yordamida aniqlanadi. Quyida *Font* metodining xususiyatlari berilgan jadvalni keltiramiz:

4-Jadval *TFont* obyektining xususiyatlari

Xususiyatlar	**Vazifasi**
Name	Shrift nomi

Size	Shrift o'lchami (punkt birligida)
Style	Belgining ko'rinishi.U oddiy,yo'g'on,yarim yo'g'on,kursiv bo'lishi mumkin. fsBold – yarim yo'g'on fsItalic – kursiv fsUnderline – tagi chizilgan fsStrikeOut – ustiga chizilgan
Color	Belgi rangi

Matnning eni va bo'yini ham ko'rsatish mumkin. Buning uchun *TextWidth* va *TextHeight* metodlaridan foydalaniladi. Quyida, forma sirtiga grafik rejimdagi matnni chiqarish dasturi keltirilgan:

```
void __fastcall TForm1::FormPaint(TObject *Sender)
{
AnsiString ms="Borland C++Builder";
TRect aRect;
int x,y;
aRect=Rect(0,0,ClientWidth,ClientHeight/2);
Canvas->Brush->Color=clWhite;
Canvas->FillRect(aRect);
aRect=Rect(0,ClientHeight/2,ClientWidth,ClientHeight);
Canvas->Brush->Color=clSkyBlue;
Canvas->FillRect(aRect);
Canvas->Font->Name="Times New Roman";
```

Canvas->Font->Size=24;

Canvas->Font->Style=TFontStyles()<<fsBold<<fsItalic;

x=(ClientWidth-Canvas->TextWidth(ms))/2;

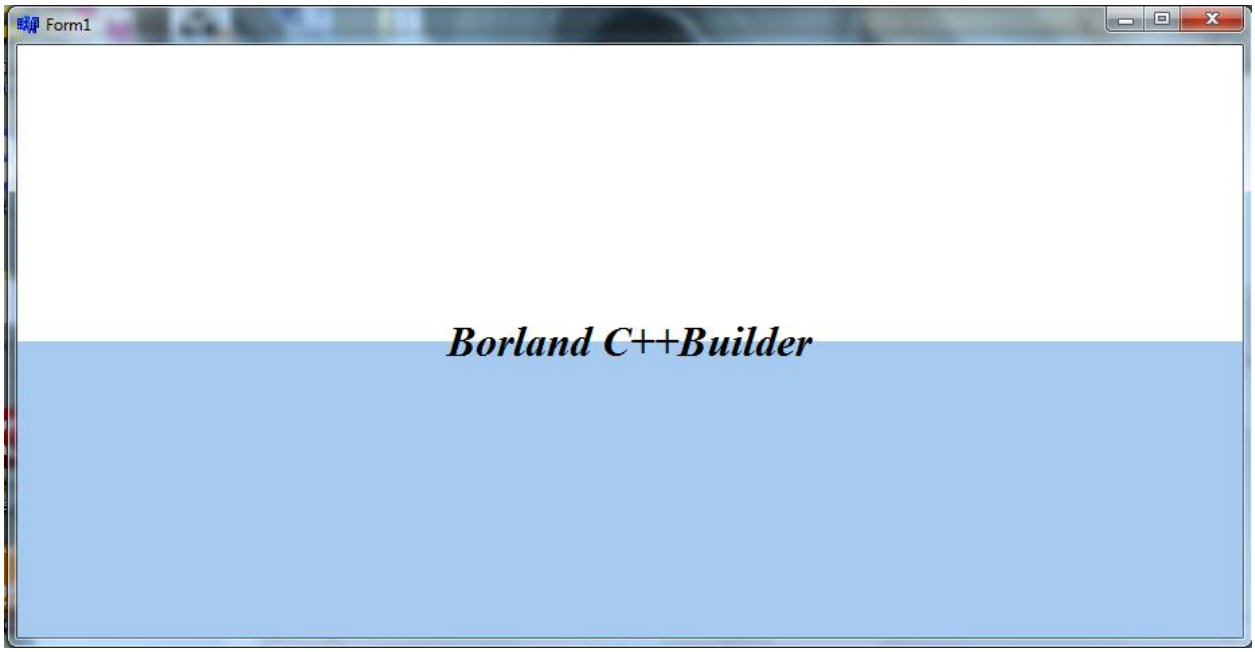
y=(ClientHeight-Canvas->TextHeight(ms))/2;

Canvas->Brush->Style=bsClear;

Canvas->Font->Color=clBlack;

Canvas->TextOutA(x,y,ms); }

Keltirilgan dastur quyidagi natijani beradi:



1.6. Rasm. Matn chiqarish

Nuqta. Grafik rejimda formaning ixtiyoriy nuqtasini belgilash mumkin. Bu jarayon belgilangan nuqtani boshqa ranga o'tkazish hisobiga amalga oshiriladi. Buning uchun *Pixels* metodidan foydalaniladi[1,12]. U *TColor* turidagi 2 o'lchovli massivni tasvirlaydi:

Canvas->Pixels[10][10]=clRed;

Bu ko’rsatma formaning *(10,10)* koordinatasini qizil rangga bo’yaydi. Nuqta o’lchami joriy grafik rejim o’lchamidan kelib chiqadi. Grafik rejimda formaning maksimal koordinatalari *ClientWidth* va *ClientHeight* bilan aniqlanadi, *Image* komponentasining maksimal koordinatalari esa *Width* va *Hieght* bilan aniqlanadi. Formaning ishchi sohasi yuqori chapdan *Pixels[0][0]*, quyi o’ngdan esa *Pixels[ClientWidth – 1][ClientHeight –1]* elementlar bilan chegaralanadi. Quyida keltirilgan dastur forma grafik o’lchamlarini avtomatik hisobga olgan holda $y = \cos^3 2x$ funksiyaning grafigini chizishni bajaradi:

```
void __fastcall TForm1::FormPaint(TObject *Sender)\
{
Grafik();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormResize(TObject *Sender)
{
TRect rct=Rect(0,0,ClientWidth,ClientHeight);
Canvas->FillRect(rct);
Grafik();
}
 #include "math.h";
float f(float x)
```

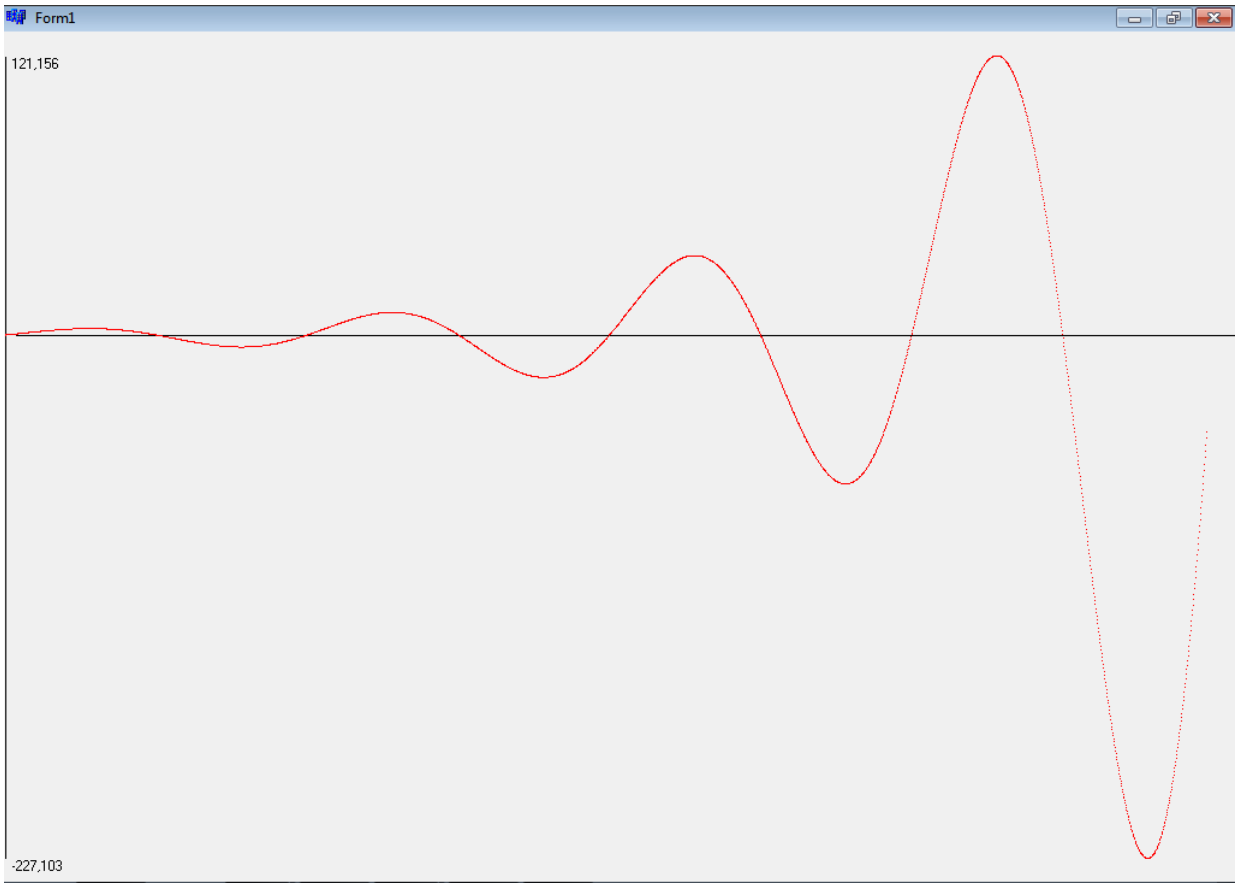
```
{ return 2*sin(x)*exp(x/5); }

void TForm1::Grafik();

{ float x1,x2;

float y1,y2;

float x;

float y;

float dx;

int l,b;

int w,h;

float mx,my;

int x0,y0;

l = 10;

b = Form1->ClientHeight-20;

h = Form1->ClientHeight-40;

w = Form1->Width-20;

x1=0;

x2=25;

dx=0.01;

x=x1;

y1=f(x);
```

```
y2=f(x);
do{
y=f(x);
if(y<y1) y1=y;
if(y>y2) y2=y;
x+=dx;
}while(x<=x2);
my=(float)h/abs(y2-y1);
mx=w/abs(x2-x1);
x0=1+abs(x1*mx);
y0=b-abs(y1*my);
Canvas->MoveTo(x0,b);Canvas->LineTo(x0,b-h);
Canvas->MoveTo(l,y0);Canvas->LineTo(l+w,y0);
Canvas->TextOutA(x0+5,b-h,FloatToStrF(y2,ffGeneral,6,3));
Canvas->TextOutA(x0+5,b,FloatToStrF(y1,ffGeneral,6,3));
x=x1;
do{
y=f(x);
Canvas->Pixels[x0+x*mx][y0-y*my]=clRed;
x+=dx;
```

}while(x<=x2);

}

Dastur izohi: Asosiy ish *Grafik* funksiyasida bajariladi. Dastlab bu funksiya berilgan funksiyaning $[x_1,x_2]$ kesmadagi minimal va maksimal qiymatlarini hisoblaydi. So'ngra mavjud ma'lumotlardan foydalanib grafik chiziladigan sohani *X* va *Y* o'qlar bo'yicha masshtablaydi. Bulardan so'ng gorizontal va vertikal o'qlarning joylashuvini hisoblaydi va chizadi va nihoyat berilgan funksiyaga mos grafik chiziladi:



1.7. Rasm. Nuqtalar asosida chizilgan grafik

Grafik funksiyasini chaqirish *OnPaint* va *OnResize* hodisalarini qayta ishlashni bajaradi.

1.2. Tasvirni tashqi fayldan o’qish

Additional komponentalar palitrasida joylashgan *Image* komponentasidan foydalanib *.bmp*, *.jpg* yoki *.ico* kengaytmali fayllardagi rasmlarni *Image* komponentasi sirtiga o’rnatish mumkin. Bu komponentaning asosiy xossalari quyidagi jadvalda keltirilgan[2,3].

5-Jadval. *Image* komponentasining xususiyatlari.

Xususiyatlar	**Vazifasi**
Picture	Komponenta sirtida rasmni aks ettirish
Width,Height	Komponenta o’lchami.Agar kompyuter o’lchami rasm o’lchamidan kichik bo’lsa, AutoSize , Strech va Proportional xususiyatlarining qiymatlari false ga o’zgaradi.
Proportional	Rasm o’lchamlarini avtomatik masshtablash belgisi
Strech	Rasmni komponentaning real o’lchamiga mos qilib avtomatik masshtablash belgisi
AutoSize	Rasmning haqiqiy o’lchamini saqlagan holda joylashtiruvchi belgi
Center	Rasmni komponenta ichida gorizontal bo’yicha pozitsiyasini aniqlovchi belgi, agar komponentaning eni rasm enidan katta bo’lsa
Visible	Komponenta yoki rasmning forma sirtiga mos kelishi yoki yo’qligini ko’rsatuvchi belgi

Canvas	Grafika chiqarilishi mumkin bo'lgan sirt

Rasmning *Image* komponentasida ko'rinishi forma ilovasiga ishlov berishda yoki dasturning bajarilishi davomida amalga oshirilishi mumkin.

Dastur ishlashi davomida rasmni komponentaga chaqirish uchun *LoadFromFile* metodidan foydalaniladi. Masalan, ushbu

Image->Picture->LoadFromFile("e.\\temp\\bart.bmp")

Ko'rsatma *bart.bmp* fayldagi tasvirni *Image1* komponentasiga o'tkazadi. Odatda, *Image* komponentasi *.bmp, .ico, .wmf* kengaytmali fayllarni o'zida tasvirlay oladi. Bu komponentada *.jpeg* kengaytmali rasmlarni tasvirlash uchun *jpeg.h* kutubxonasini chaqirish kerak. Agar bu kutubxona chaqirilmagan bo'lsa, dastur bajarilishi mobaynida *jpeg* fayl chaqirilganda xatolik beradi.

Quyida dastur *Image* komponentasida *.jpg* kengaytmali rasmni tasvirlashni amalga oshiradi. "Katalog" tugmasi rasm faylini kataloglar ichidan topib tanlash imkonini beradi. "Keyingi" tugmasi navbatdagi rasm faylini ko'rsatish imkonini beradi

```
#include <jpeg.hpp>
ffinclude <FileCtrl .hpp>
AnsiString aPath;
TSearchRec aSearchRec;
void _ fastcall TForml: : FormCreate (TObject *Sender)
{
aPath = "";
Imagel->AutoSize = false;
Imagel->Proportional = true;
Button2->Enabled = false;
FirstPicture () ;
```

```
}
void _ fastcall TForml: :ButtonlClick (TObject ^Sender)
{
if (SelectDirectory (
"Tasvir joylashgan papkani tanlang",
"",aPath) != 0)
aPath = aPath + "\\";
FirstPicture ( ) ;
}
void TForml : : FirstPicture ( )
{
Imagel->Visible = false;
Button2->Enabled = false;
Labell->Caption = "";
if ( FindFirst (aPath+ "*.jpg", faAnyFile, aSearchRec) == 0)
Imagel->Picture->LoadFromFile(aPath+aSearchRec.Name);
Imagel->Visible = true;
Labell->Caption = aSearchRec.Name;
if ( FindNext(aSearchRec) ==0)
{
Button2->Enabled = true;
}
void _ fastcall TForrol: :Button2Click (TObject *Sender)
{
Imagel->Picture->LoadFromFile (aPath+aSearchRec.Name) ;
Labell->Caption = aSearchRec.Name;
if ( FindNext (aSearchRec) != 0) {
Button2->Enabled = false; }
```

}

Birinchi va qolgan rasm fayllarini yuklash uchun mos ravishda *FirstPicture* va *NextPicture* funksiyalaridan foydalaniladi. *FirstPicture* funksiyasi tasvirlanishi lozim bo'lgan fayl nomini o'zlashtirish uchun *FindFirst* funksiyasini chaqiradi. *FindFirst* funksiyasining parametri sifatida quyidagilar uzatiladi:

- Rasm fayli joylashgan katalog nomi;
- Qidiruv kriteriyasini qanoatlantiruvchi *SearchRec* strukturali fayllar nomi;'
- Rasm mfaylining maskasi.

Agar katalogda izlanayotgan kriteriyaga mos bitta fayl topilsa ham, *FindFirst* funksiyasi *0* qiymatni qaytaradi. Bu holatda *LoadFromFile* funksiyasi rasmni forma yoki komponenta sirtiga chiqaradi. 1-rasm chaqirilgandan so'ng *FirstPicture* funsiyasi *FindNext* funksiyasini chaqiradi. Agar fayl topilsa, "Keyingi" tugmasi faollashadi.

Bundan so'ng "Keyingi" tugmasiga ta'sir etib, navbatdagi rasmlarni tasvirlash mumkin. Agar navbatadgi rasmlar topilmasa "Keyingi" tugmasi faollashmaydi. Rasm o'lchamlarini o'zgarishsiz chiqarish uchun yuqoridagi *Image* komponentasi belgilaridan foydalanish kerak bo'ladi. Masalan:

Image1->AutoSize=false;

Image1->Proportional=true;

1.3. Bitli tasvirlardan foydalanish

Murakkab tasvirlarni tashkil etish uchun bitli tasvirlashdan foydalaniladi. **Bitli tasvir**—kompyuter xotirasida joylashgan katta bo'lmagan rasmdir. Bitli tasvirni baytlar yoki fayldagi resurslar orqali tashkil etish mumkin. Shuningdek, "forma sirtidan yoki boshqa bitli tasvirlardan olingan nusxalar bo'lishi mumkin"[2,12].

Bitli tasvirlarni garfik muharrirlar yordamida tayyorlash yoki dasturiy vositalar yordamida resurslardan yuklab olish mumkin. Oxirgi holat fayl resurslarini tashkil etish va unga bitli tasvirlarni joylashtirish imkonini beradi. Fayl resurslarini *ImageEditor* utilitasi yordamida ham yaratish va tahrirlash mumkin.

Dasturdagi bitli tasvir — bu *TBitmap* obyektidir. Quyidagi jadvalda *TBitmap* ning ayrim xossalari keltirilgan:

6-Jadval. *TBitmap* obyektining xususiyatlari

Xususiyatlari	Vazifasi
Height,Width	Bitli tasvir o'lchamlari. Fayl yoki resursdan yuklangan bitli tasvirga mos keluvchi qiymatlar
Empty	Bitli tasvirning fayldan yuklanganlik belgisi
Transparent	Shaffof rangni o'rnatish. Bu rang bilan bo'yalgan rasm elementlari Draw metodi chaqirilmaydi
Transparent Color	Shaffof rang rejimini o'rnatish. Odatda TransParentColor quyi chap piksel rangi bilan aniqlanadi
Canvas	Bitli tasvir sirti . Rasm xuddi forma sirtida yoki Image komponentasida chizilgani kabi

Bitli tasvirni bmp fayllarni chaqirilgani kabi *LoadFromFile* metodi bilan chaqirish mumkin. Masalan, quyidagi dastur bo'lagi bitli tasvirni fayldan yuklash ishini bajaradi:

```
#include <vcl.h>
```

```
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
Graphics::TBitmap *Background, *Bitmap, *Buf;
int W,H,x,y;
TRect BufRect,BackRot;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 Background = new Graphics::TBitmap();
Bitmap = new Graphics::TBitmap();
Buf = new Graphics::TBitmap();
Background->LoadFromFile("sea.bmp");
```

```
Form1->Image1->Canvas->Draw(0,0,Background);

Bitmap->LoadFromFile("ship.bmp");

Bitmap->Transparent = true;

Bitmap->TransparentColor = Bitmap->Canvas->Pixels[1][1];

W= Bitmap->Width;

H= Bitmap->Height;

Buf->Width= W;

Buf->Height=H;

Buf->Palette=Background->Palette;

Buf->Canvas->CopyMode=cmSrcCopy;

BufRect=Bounds(0,0,W,H);

x = -150;

y = 100;

BackRot=Bounds(x,y,W,H);

Buf->Canvas->CopyRect(BufRect,Background->Canvas,BackRot);

Form1->DoubleBuffered = true; }

//---------------------------------------------------------------------------

void __fastcall TForm1::Timer1Timer(TObject *Sender)

{  Form1->Image1->Canvas->Draw(x,y,Buf);

x++;
```

if (x>Form1->Image1->Width) x=-W;

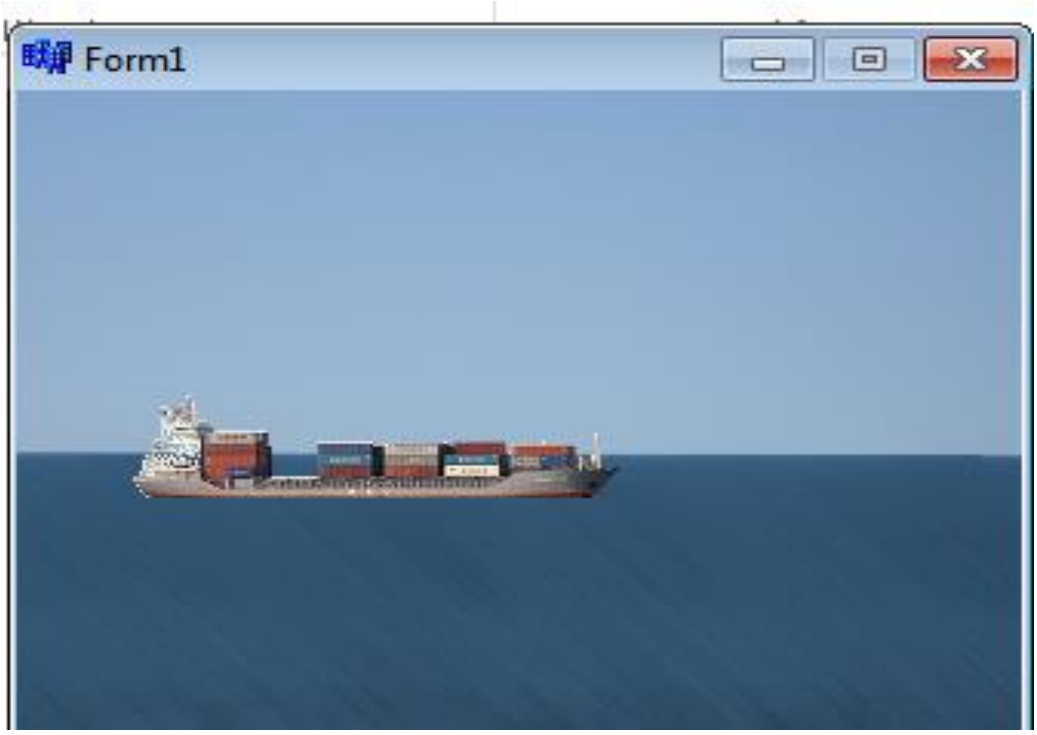
BackRot=Bounds(x,y,W,H);

Buf->Canvas->CopyRect(BufRect,Background->Canvas,BackRot);

Form1->Image1->Canvas->Draw(x,y,Bitmap);

Timer1->Interval=50;}

Ushbu dastur ishlashi natijasida forma oynasida quyidagi harakatlanuvchi tasvir hosil bo'ladi:



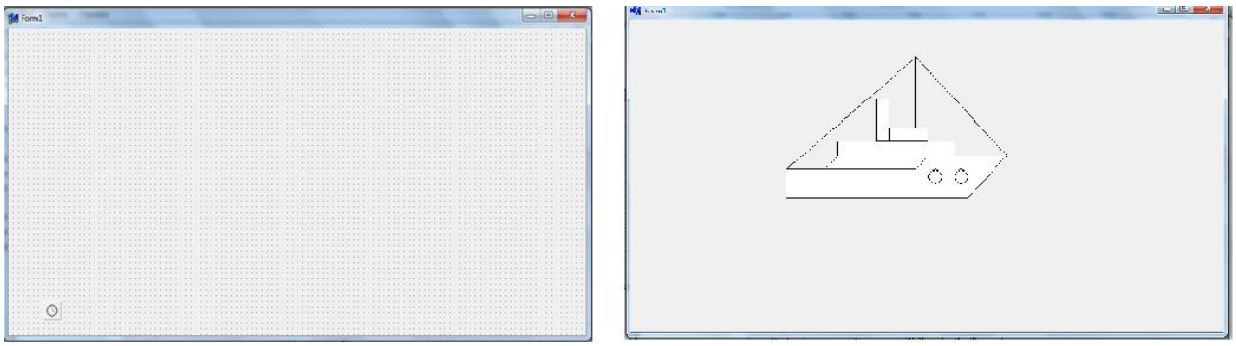
1.8.Rasm. Harakatlanuvchi kema tasviri

II BOB MULTIPLIKATSIYA VA MULTIMEDIA

2.1. Multiplikatsiya

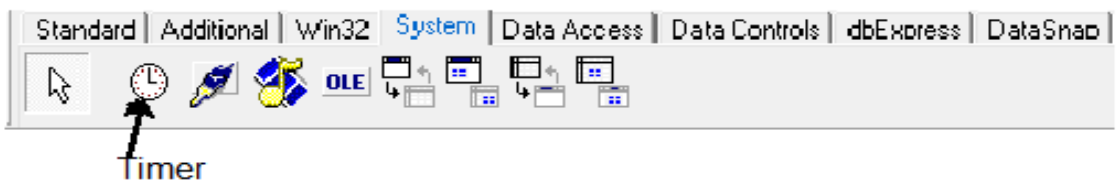
Multiplikatsiya – bu harakatlanuvchi va almashinuvchi rasmlardir. Oddiy holatda rasm siljiydi yoki almashinadi. Rasmni siljitish yetarlicha oddiy: dastlab rasmni ekranga chiqarish kerak bo'ladi, so'ngra uni o'chirish va avvalgisidan o'zgargan masofada yana ekranga chiqarish kerak bo'ladi. Chiquvchi va o'chiruvchi hamda yangi joydan paydo bo'luvchi rasmlar vaqt oralig'i bir joyda yi'g'ilib tasvirlanishi kerak[1].

Bazaviy nuqta metodi. Quyidagi (grafik primitivlardan tuzilgan (1)) dastur rasmni osonlik bilan siljitishni bajaradi. Oyna va formaning ko'rinishi 2.1. Rasmda keltirilgan.



2.1. Rasm. Dastur oynasi va forrmasining ko'rinishi

Forma sirtida yagona *Timer* komponentasi joylashgan, u o'giriluvchi va hosil qilinuvchi tasvirlar orasidagi vaqtni boshqaradi. Bu komponenta *System* komponentalar palitrasida joylashgan. Shuningdek bu komponenta novizual hisoblanadi.



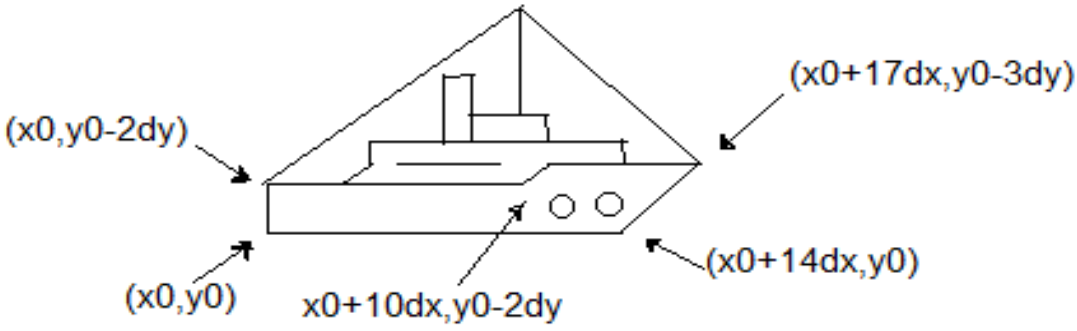
2.2. Rasm. Timer komponentasining joylashuv o'rni

Quyida *Timer* komponentasining xossalarini keltiramiz:

7-Jadval. Timer komponentasining xossalari.

Xossasi	Ta’rifi
Name	Komponenta nomi.Komponenta xossasiga murojaat ruxsatini berish
Interval	OnTimer hodisasining kechishi. Millisekundlarda beriladi
Enabled	Hodisa boshlanishiga ruxsat berish yoki bermaslik

Timer komponentasi *OnTimer* hodisasini generatsiya qiladi. *Enabled* xossasiga diqqatni qaratish zarur. U dasturda hodisaning ro’y berish yoki bermasligini aniqlaydi, ya’ni agar uning qiymati false bo’lsa, hodisa ro’y bermaydi. Keltiriladigan dasturda kema rasmini hosil qilish uchun Ship funksiyasidan foydalaniladi. *Ship* funksiyasining parametri sifatida bazaviy nuqta beriladi. Bazaviy nuqta koordinatalari (x_0, y_0) bo’lib, ular butun sonlarda beriladi, siljish qadami esa ma’lum birlikka ega bo’ladi.



2.3.Rasm. Bazaviy nuqtasi *(x_0, y_0)* bo’lgan kema tasviri

Kemaning yangi tasviri chizilishidan oldin *OnTimer* dan oldingi chizilgan kema tasviri o'giriladi, ya'ni u o'ziga mos to'g'ri to'rtburchakli soha bilan yopiladi. Dastur matnini quyida keltiramiz:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int x=-340,y=250;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{ Canvas->Brush->Color=Form1->Color;
Canvas->FillRect(Rect(x-1,y+1,x+340,y-250));
x+=3;
```

```
if(x>ClientWidth)
{   x=-340;
y=random(Form1->ClientHeight); }
int dx=20,dy=20;
Canvas->Pen->Color=clBlack;
Canvas->Brush->Color=clWhite;
TPoint p1[7];
p1[0]=Point(x,y);
p1[1]=Point(x,y-2*dy);
p1[2]=Point(x+10*dx,y-2*dy);
p1[3]=Point(x+11*dx,y-3*dy);
p1[4]=Point(x+17*dx,y-3*dy);
p1[5]=Point(x+14*dx,y);
p1[6]=Point(x,y);
Canvas->Polygon(p1,6);
TPoint p2[8];
p2[0]=Point(x+3*dx,y-2*dy);
p2[1]=Point(x+4*dx,y-3*dy);
p2[2]=Point(x+4*dx,y-4*dy);
p2[3]=Point(x+13*dx,y-4*dy);
```

```
p2[4]=Point(x+13*dx,y-3*dy);

p2[5]=Point(x+11*dx,y-3*dy);

p2[6]=Point(x+10*dx,y-2*dy);

p2[7]=Point(x+3*dx,y-2*dy);

Canvas->Polygon(p2,7);

Canvas->MoveTo(x+5*dx,y-3*dy);

Canvas->LineTo(x+9*dx,y-3*dy);

Canvas->Rectangle(x+8*dx,y-4*dy,x+11*dx,y-5*dy);

Canvas->Rectangle(x+7*dx,y-4*dy,x+8*dx,y-7*dy);

Canvas->Ellipse(x+11*dx,y-2*dy,x+12*dx,y-1*dy);

Canvas->Ellipse(x+13*dx,y-2*dy,x+14*dx,y-1*dy);

Canvas->MoveTo(x+10*dx,y-5*dy);

Canvas->LineTo(x+10*dx,y-10*dy);

Canvas->MoveTo(x+17*dx,y-3*dy);

Canvas->LineTo(x+10*dx,y-10*dy);

Canvas->LineTo(x,y-2*dy);

Timer1->Interval=50;

}
```

2.2. Multimediali fayllar bilan ishlashda Animate komponentasining qo'llanilishi

Windows muhitida ishlovchi ko'plab dasturiy vositalar mutimediali hisoblanadi. Bunda dsaturlar videorolik va multiplikatsiya, ovozli fayllarni ishlatishda qo'llaniladi. Multimediali dasturlarga o'yin va o'rgatuvchi dasturlarni misol qilib ko'rsatish mumkin. C++ Builder muhiti multimediali fayllar bilan ishlashga 2 ta komponentani taqdim etadi[1,2,12]. Ular quyidagilar:

- Animate – oddiy ovozsiz animatsiya bilan ishlovchi komponenta ;
- MediaPlayer – viderolik, ovoz va ovozli tasvirlarni boshqaruvchi komponenta .

Animate komponentasi *Win32* komponentalar guruhida joylashgan bo'lib, **.avi* faylidagi ovozsiz animatsiyani boshqaradi.

2.4-rasm. Animate komponentasi belgisi

Bu komponenta formaga oddiy holatda qo'shiladi. Komponenta formaga joylashtirilgach uning xossalari joylanadi. Quyidagi jadvalda *Animate* komponentasining xossalari keltrilgan[15]:

8-Jadval. *Animate* komponentasining xossalari

Xossasi	Ta'rifi
Name	Komponenta nomi.Komponenta xossalariga murojaat va uning holatini boshqarish
FileName	Komponenta yoradamida boshqariladigan .avi fayl nomi
FrameWidth	Animatsiya kadrining kengligi

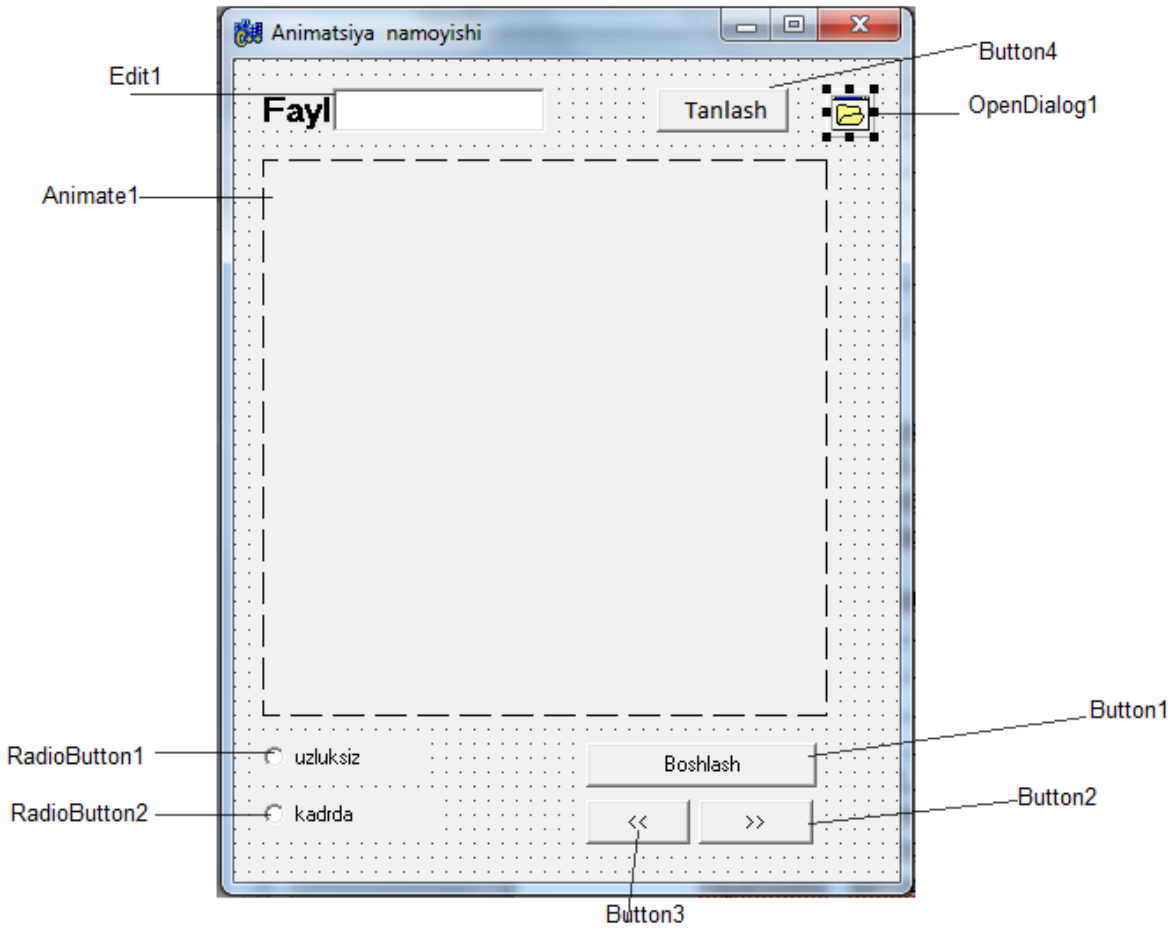
FrameHeight	Animatsiya kadrining balandligi
FrameCount	Animatsiyadagi kadrlar soni
AutoSize	Animatsiya kadrining o'lchamiga qarab, komponenta o'lchamini o'zgartirish
Center	Komponenta maydonida animatsiya kadrini markazlashtirish.Agar uning qiymati true bo'lsa va kadr o'lchami komponenta o'lchamidan katta bo'lsa, u markazlashtriladi
StartFrame	Animatsiya boshlanishi kerak bo'lgan kadr nomeri
StopFrame	Animatsiya yakunlanishi kerak bo'lgan kadr nomeri
Active	Animatsiyani aks ettrishni faollashtirish belgisi
Color	Animatsiya jarayonidagi fon rangi
Transparent	Animatsiya aks etishida shaffof rang rejimi
Repetitions	Animatsiyani takroriy aks ettirishlar soni
CommonAVI	Windowsning standart animatsiyasini aniqlash

Animate komponentasi dasturchiga o'z dasturida standart animatsiyalardan foydalanish imkonini beradi. Animatsiya ko'rinishini *CommonAvi* xossasining qiymati aniqlaydi. Qiymat nomlanga konstantalar yordamida beriladi. Jadvalda animatsiya namoyishida foydalaniladigan bir qator konstantalar keltrilgan:

9-Jadval. *CommonAvi* xossasini aniqlovchilar mazmuni

Mazmun	Animatsiya	Jarayon
aviCopyFiles		Fayllarni nusxalash
aviDeleteFile		Faylni o’chirish
aviRecycyleFile		Faylni savatchadan o’chirish

Shuni yana bir bor ta’kidlash lozimki, Animate komponentasi AVI faylda joylashgan sof animatsiyani boshqaradi. Agar ovozli animatsiya komponentaga bog’langa bo’lsa, dastur bajarilishi davomida xatolik ro’y beradi. Quyida keltirilgan dastur animatsiyani namoyish etishga mo’ljallangan:



2.5.Rasm. “Animatsiya namoyishi” dasturi formasi

Dastur ishga tushirilgach loyiha joylashgan katalogdagi animatsiya fayllarining dastlabki kadri formaga ko’rinadi. Agar joriy katalogda birorta ham *.*avi* kengaytmali fayllar bo’lmasa, formadagi Anamate komponentasi bo’sh qoladi.

Animatsiyali fayl nomini *Edit1* komponentasiga kiritish yoki “Tanlash” tugmasi yordamida standart muloqot oynasini ochish mumkin. Fayllarni ochishning standart muloqot oynasidan foydalanishga *OpenDialog1* komponentasi yordam beradi. Bu komponenta *Dialogs* komponentalar palitrasida joylashgan.

“Animatsiya namoyishi” dasturi uni ko’rishning 2 ta rejimini ta’minlaydi: uzluksiz va kadrlar bo’yicha o’tkazish.

Button1 tugmasi animatsiyani boshlash va uni to’xtatish uchun xizmat qiladi. Animatsiyani uzluksiz namoyish etish jarayoni uchun “Pusk” tugmasiga *OnClick* hodisasini qo’shish va *Active* xossasiga *true* qiymatni berish kerak bo’ladi. Bu protsedura *Button1* tugmasidagi *Pusk* nomini *Stop* bilan almashtiradi. Animatsiya rejimlari *RadioButton1* va *RadioButton2* tugmalari yordamida almashtiriladi. *OnClick* hodisasini qayta ishlovchi protsedura bu komponentalarda *Enabled* xossasini o’zgartirish “bloklash” yoki “ochish” ni bajaradi. Kadrlarni o’tkazish rejimi uchun mo’ljallangan “oldingi” (Button2) , “keyingi” (Button3) tugmalarini faollashtiradi. Uzluksiz animatsiya jarayonini to’xtatilishida Stop tugmasining *Active* xossasi *false* qiymatini oladi.

```
#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart_init)

#pragma resource "*.dfm"

TForm1 *Form1;

int CFrame;
```

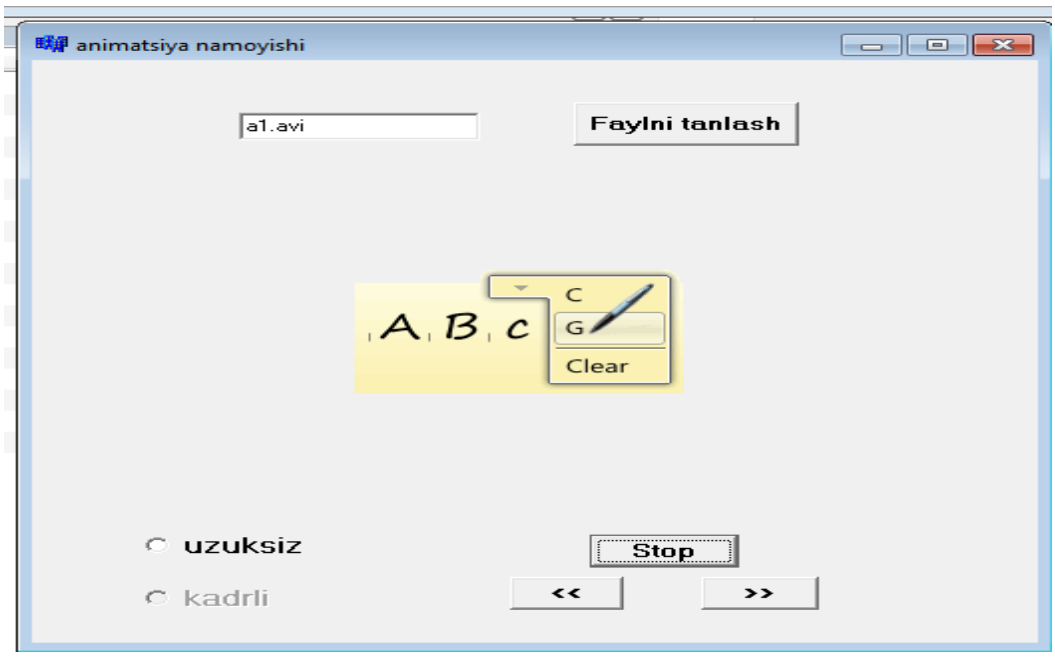
```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{   TSearchRec sr;
if(FindFirst("*.avi",faAnyFile,sr)==0)
{  Edit1->Text=sr.Name;
try
{  Animate1->FileName=sr.Name;  }
catch(Exception &e)
{  return;
}
RadioButton1->Enabled=true;
RadioButton2->Enabled=true;
Button1->Enabled=true;   }  }
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
```

```
{  OpenDialog1->InitialDir="";
OpenDialog1->FileName="*.avi";
if(OpenDialog1->Execute())
{  try
{ Animate1->FileName=OpenDialog1->FileName;  }
catch(Exception &e)
{  Edit1->Text="";
AnsiString msg="faylni  ochishda xatolik"+OpenDialog1->FileName+"\n ovoz soprovoj animatsiya bo'lishi mumkin";
ShowMessage(msg);
return;  }
Edit1->Text=OpenDialog1->FileName;
RadioButton1->Enabled=true;
RadioButton2->Enabled=true;  }  }
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{  if(Animate1->Active)
{  Animate1->Active=false;
Button1->Caption="Boshlash";
RadioButton2->Enabled=true;  }
else  {
```

```
Animate1->StartFrame=1;

Animate1->StopFrame=Animate1->FrameCount;

Animate1->Active=true;

Button1->Caption="Stop";

RadioButton2->Enabled=false;  }  }

//---------------------------------------------------------------------------

void __fastcall TForm1::RadioButton1Click(TObject *Sender)

{  Button1->Enabled=true;

Button2->Enabled=false;

Button3->Enabled=false;

Animate1->Active=false;  }

//---------------------------------------------------------------------------

void __fastcall TForm1::RadioButton2Click(TObject *Sender)

{  Button1->Enabled=false;

Button2->Enabled=true;

Button3->Enabled=false;

Animate1->StartFrame=1;

Animate1->StopFrame==1;

Animate1->Active=true;

CFrame = 1; }
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{   CFrame++;
Animate1->StartFrame=CFrame;
Animate1->StopFrame==CFrame;
Animate1->Active=true;
if(CFrame>1)
Button3->Enabled=true;
if(CFrame==Animate1->FrameCount)
Button2->Enabled=false;}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{  if(CFrame==Animate1->FrameCount)
Button2->Enabled=true;
CFrame--;
Animate1->StartFrame=CFrame;
Animate1->StopFrame=CFrame;
Animate1->Active=true;
if(CFrame==1)
Button3->Enabled=false;}
```

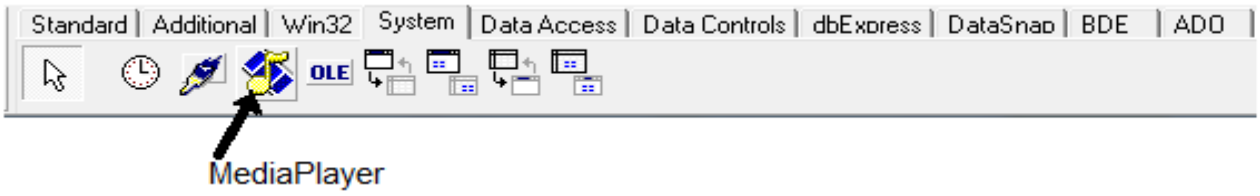
Keltirilgan dastur ishlashi natijasida ko’zlangan maqsadga erishildi va natijalar olindi:



2.6.Rasm. *Animate* komponentasidan foydalanib tuzilgan dastur natijasi

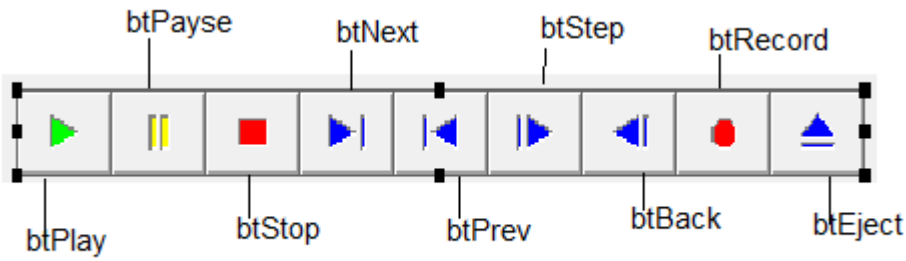
2.3. Multimediali fayllar bilan ishlashda MediaPlayer komponentasining qo’llanilishi

MediaPlayer komponentasi bir qator ovozli fayllarni (WAV,MID,RMI,MP3) boshqarish imkonini beradi. Bu komponenta *System* komponentalar palitrasida joylashgan[2,16].



2.7.Rasm.*MediaPlayer* komponentasining belgisi.

Bu komponenta oddiy video yoki audio playerda foydalaniladigan bir qator tugmalarni o'zida saqlaydi. Bu tugmalarning vazifalari quyidagi jadvalda keltirilgan:



2.8.Rasm *MediaPlayer* komponentasining forma sirtidagi ko'rinishi

Quyidagi jadvalda *MediaPlayer* komponentasining formaga joylashtirilgan holatida faollashadigan xossalari keltirilgan :

10-Jadval *MediaPlayer* komponentasining xossalari

Xossasi	Vazifasi
Name	Komponenta nomi. Ishchi holatda kompyuterning mavjud xossalari
DeviceType	Qurilma turi.MediaPlayer komponentasi uchun aniq qurilmani belgilaydi.Qurilma turi nomlangan konstantalar bilan beriladi: dtAutoSelect – fayllar kengaytmasi bo'yicha turni avtomatik aniqlash, dtVaweAudio – ovoz chiqarish uskuna,dtAviVideo – videoni ko'rish uskunasi, dtCDAudio – CD uskunasi
FileName	Namoyish etiladigan videorolik yoki ovozli fayl nomi

AutoOpen	Ovozli fayl yoki videorolikni dastur ishga tushishi bilan avtomatik yuklash belgisi
Display	Videorolikni namoyish etuvchi ekran ko’rinishini aniqlash(oddiy sifatdagi video uchun Panel komponentasidan foydalaniladi)
Visible Buttons	Tarkibiy xossa.Komponentaning ko’rinadigan tugmalarini aniqlaydi.Ayrim tugmalarning ko’rinmas bo’lishini ta’minlaydi.

Komponentaning formadagi ishchi holatida faol bo’lgan xossalar dastur ishlashi davomida ham faol bo’ladi va bular namoyish jarayonidagi bir qator ma’lumotlarni olish imkonini beradi.Davomiylik haqidagi ma’lumotni saqlovchi xossa turli formatda tasvirlanishi mumkin. Bular orasidan universal format sifatida tfMilliseconds formatini olish mumkin. Bunda davomiylik millisekundlarda ifodalanadi. Ayrim qurilmalar bir qancha qurilmalarni qo’llab-quvvatlaydi. (Masalan,tfTMSF formati). Millisekundlarni minut va sekundlar yordamida ifodalash uchun ma’lum munosabatlardan foydalaniladi. Agar xossa tfTMSF formatida bo’lsa, u holda akslantirish MCI_TMSF_TRACK, MCI_TMSF_SECOND VA MCI_TMSF_MINUTE makroslari bilan amlga oshiriladi. Bu kabi foydali makroslarni mmsystem.h sarlavha faylidan olish mumkin.

11-Jadval. Dastur ishlashi jarayonida faol bo’lgan *MediaPlayer* komponentasining xossalari.

Xossalari	**Vazifasi**

Length	Ochilgan fayl yoki audio CD ning uzunligi(ijro etish vaqti)
Tracks	Ochilgan qurilmadagi treklar soni
TrackLength	Trek davomiyligi.Bu xossa o'zida massivni tasvirlaydi
Position	Namoyish boshlanishidagi trek holati
TimeFormat	Legth, TrackLength va Position xossalarining qiymatini tasvirlovchi format.tfMilliseconds – universal format.Agar MediaPlayer ovozli CDdan foydalanayotgan bo'lsa, u holda tfTMSF formatidan foydalanish kerak
Mode	Ijro qurilmasining holati. mpPlaying – qurilma ijro holatida, mpStopped – qurilma to'xtagan holatda, mpPaused – vaqtinchalik to'xtagan holatda, mpNotReady – qurilma ishlashga tayyor emas, mpOpen – qurilmada fayl saqlovchisi yo'q
Display	Klipni namoyish etishga mo'ljallangan soha. Agar bu xossa ko'rsatilmagan bo'lsa, tasvir dastur bajarilishida alohida oynad chiqadi.
DisplayRect	Ekran sohasida klipni tasvirlash o'lchamlari

MediaPlayer komponentasi dastur ishlashida foydalanuvchi uchun bir qator qulay metodlarni tavsiya etadi[15]. Quyidagi jadvalda bu metodlar keltirilgan:

12-Jadval *MediaPlayer* komponentasining metodlari.

Metod	Vazifasi
Play()	Ijro holatini faollashtirish.Play tugmasiga ta’sir etishga o’xshash metod
Stop()	Ijro holatini to’xtatish
Pause()	Ijro holatini vaqtinchalik to’xtatish
Next()	Navbatdagi trekka o’tish
Previous()	Oldingi trekka o’tish
Step()	Navbatdagi kadrga o’tish
Back()	Oldingi kadrga o’tish

III BOB. C++ TILI MULTIMEDIA IMKONIYATLARINING TATBIG'I

3.1. Bir nechta funksiya grafiklarini bir vaqtda tasvirlash

Hozirgi kunda ko'plab dasturiy paketlar mavjud bo'lib, ular yordamida hisob kitob, grafikaga oid ishlarni osonlik bilan bajarish mumkin. Shunga qaramasdan bir qator sohalarda uchraydigan tipik masalalar uchun alohida yondashuvga ehtiyoj seziladi. Chunki bu masalalarni hal etishda dasturiy paketlarda mavjud imkoniyatkar yetarli bo'lmaydi. Shu boisdan ham zamonaviy dasturlash tillarida mavjud hisob kitob, grafik VA multimedia imkoniyatlardan foydalanishni mukammal o'rganish bo'lajak dasturchilar oldida dolzarb bo'lgan masalalardan biridir.

Bularni hisobga olib biz quyida C++ Builder dasturlash tilining grafik imkoniyatlaridan foydalanib funksiya grafiklarini chizish jarayonini bayon etamiz. Shu maqsadda bir nechta funksiyalarning grafik tasvirini o'zida aks ettiruvchi dastur qanday yaratilishini ko'rib o'tamiz. Buning uchun quyidagi amallar ketma ketligi bajariladi:

1. C++ Builder dasturini ishga tushiramiz va yangi forma ilovasini yaratamiz.
2. Loyiha nomini o'zgartirish uchun *Form1* ning *Caption* xususiyatini o'zgartiramiz.
3. Additional komponentalar bo'limidan Chart komponentasini formaga joylashtiramiz. Bu komponenta har xil turdagi grafik va diagrammalar yaratish uchun qo'llaniladi.
4. Endi *Chart* komponentasining xususiyatlarini o'rnatamiz. *Align* xususiyatining qiymatini *allClient* ga o'zgartiramiz. Bu orqali tasvirlanadigan grafikni formaning butun tekisligiga joylashtiramiz.
5. *Chart* komponentasi ustiga sichqoncha tugmasini ikki marta tez-tez bosish orqali *"Диаграмм Chart"* oynasiga o'tamiz. Bu oyna ikkita asosiy sahifani o'z ichiga oladi: *Chart* va *Series*. *Add* tugmasi yangi seriya qo'shish imkoniyatini beradi. Bu tugmani bosganda grafik yoki diagrammaning tipini

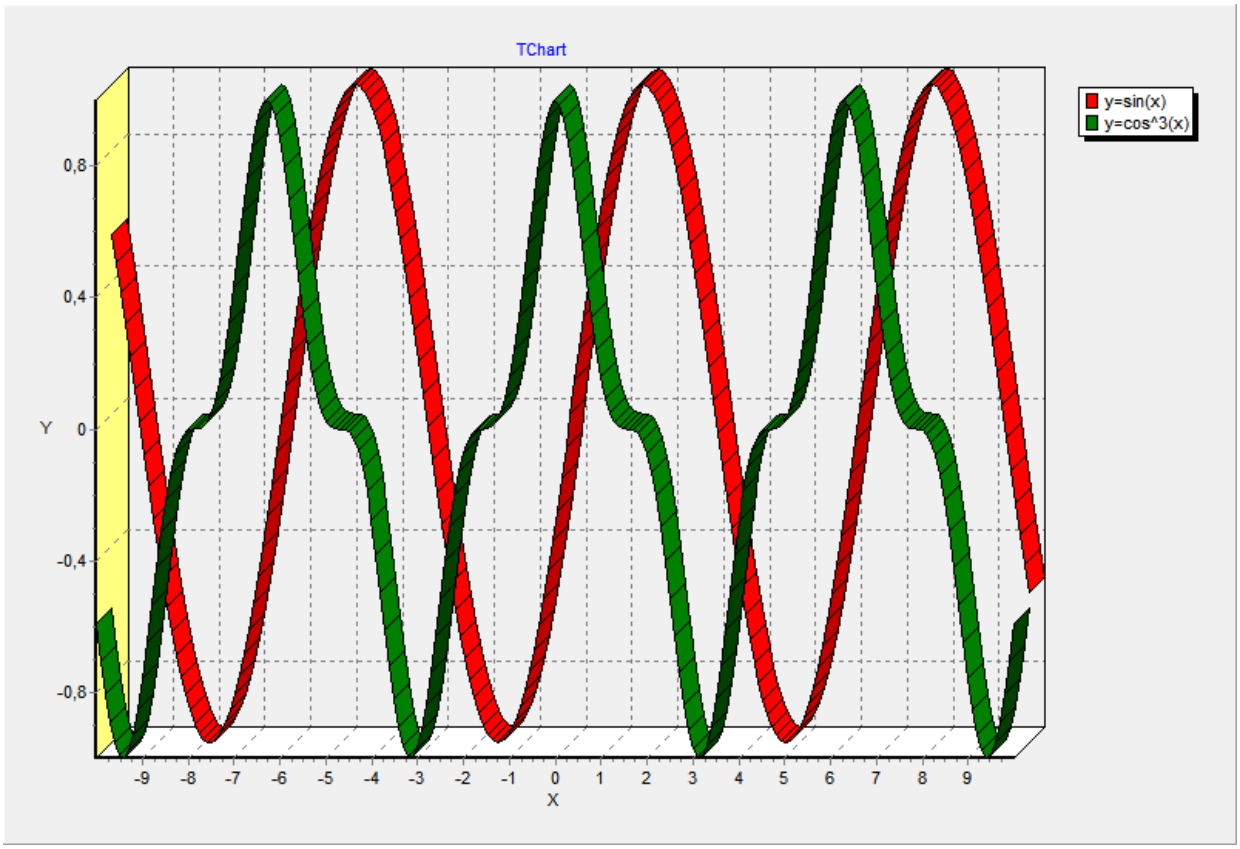
tanlash oynasi ochiladi. Bizning misolimizda *Line* tipi tanlanadi. Bu misolda ikkita funksiyaning grafigini qurish uchun 2 ta seriya yaratiladi.

6. *Title* tugmasi yordamida yangi seriyalarga nom o'zlashtiriladi. Birinchi seriya *y=sin(x)*, ikkinchi seriya *y=cos^3(x)* ko'rinishda nomlanadi.
7. *Chart* sahifasidagi *Titles* orqali *Chart* obyektiga nom berish va uning xususiyatlarini o'rnatish mumkin.
8. *Axis* qismida grafik o'qlarining parametrlari sozlanadi. Bunda chap vertikal o'q parametrlarini sozlash uchun *Left* tanlanadi. Unga nom o'zlashtirish uchun *Title* qismiga o'tiladi va kerakli nom yoziladi. Yozuvni gorizontal holatga keltirish uchun *Angle* maydoniga *0* qiymati kiritiladi. *Labels* qismida yozuv stilini *Value* ga o'zgartiramiz. Pastki gorizontal o'q *(Bottom)* parametrlari ham shu tarzda sozlanadi.
9. Grafikda seriyalar nomini aks ettirish uchun *Legend* qismiga o'tamiz va *Legend Style* maydonining qiymatini *Series Names* ga o'zgartiramiz.
10. Endi asosiy dastur matnini kiritish uchun *Form1* ning *Events* hodisalar oynasiga o'tamiz va *OnActivate* maydonini faollashtiramiz. Hosil bo'lgan oynaga quyidagi dastur matnini kiritamiz:

```
void __fastcall TForm1::FormActivate(TObject *Sender)

{

double x,y,y1;

Series1->Clear();

Series2->Clear();


for (x=-10;x<=10;x=x+0.1)

{
```

```
y=sin(x);

y1=pow(cos(x),3);

Series1->AddXY(x,y);

Series2->AddXY(x,y1);

}

Chart1->Refresh();

Chart1->LeftAxis->Increment=(Chart1->LeftAxis->Maximum-Chart1->LeftAxis->Minimum)/5;

}
```

Ushbu dastur ishlashi natijasida quyidagi natija olinadi:

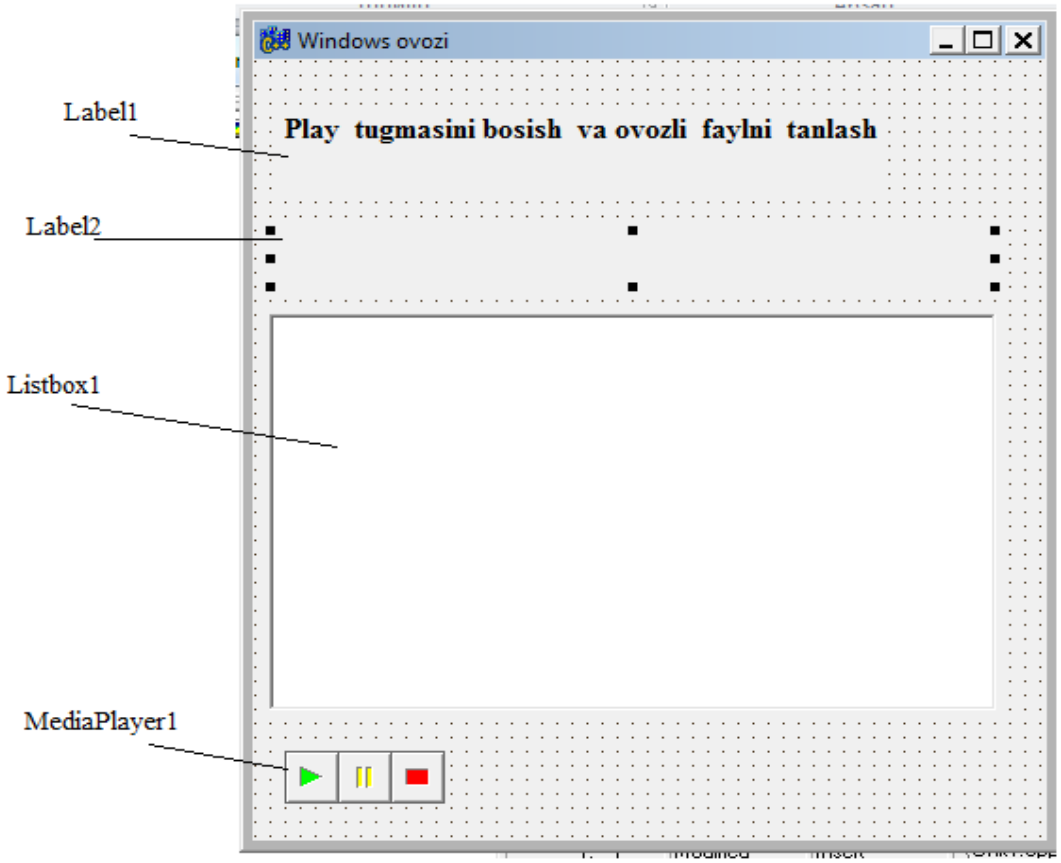


3.1. Rasm. *Chart* komponentasi yordamida yaratilgan dastur natijasi

Ko’rib o’tilgan dastur loyihasi C++ Builderning grafik imkoniyatlaridan biri bo’lib, masalaning qo’yilishiga qarab uning boshqa imkoniyatlarini ham ishga qo’shish mumkin. Bundan tashqari grafik imkoniyatlar bilan hisob kitoblarga bog’liq imkoniyallarni birga ishlatish aniqligi yuqori bo’lgan grafiklarni olish, kompyuterga ulangan turli qurilmalardan kelayotgan signallarni barchaga tushunarli tarzda tasvirlash imkonini beradi.

3.2. Ovozli fayllar bilan ishlash

MediaPlayer komponentasidan foydalanib faqat ovozdan iborat faylni ijro ettirish mumkin. Bu borada Windows tomonidan boshqariladigan ovozli fayl fragmentini qaraymiz. 3.2.-rasmda ovozli fragmentni boshqaruvchi forma ko’rinishi tasvirlangan.



3.2. Rasm. Ovozli fayllar bilan ishlovchi dastur formasining ko’rinishi

Ovozli fayllar bilan ishlovchi *MediaPlayer* komponentasining qo'llanilgan xossalari:

13-Jadval *MediaPlayer* komponentasi xossalarining mazmuni

Komponenta	Mazmuni
DeviceType	dtAutoSelect
VisibleButtons.btNext	False
VisibleButtons.btPrev	False
VisibleButtons.btStep	False
VisibleButtons.btBack	False
VisibleButtons.btRecord	False
VisibleButtons.btEject	False

Formada *MediaPlayer* komponentasi bilan birga *ListBox* komponentasi ham o'rnatilgan. Undan foydalanib ijro etiladigan ovozli faylni tanlash mumkin. Shuningdek, 2 ta *Label* komponentasi ham formada joylashgan. Birinchisi xabar chiqarish uchun, ikkinchisi esa tanlangan fayl nomini aks ettirish uchun xizmat qiladi.

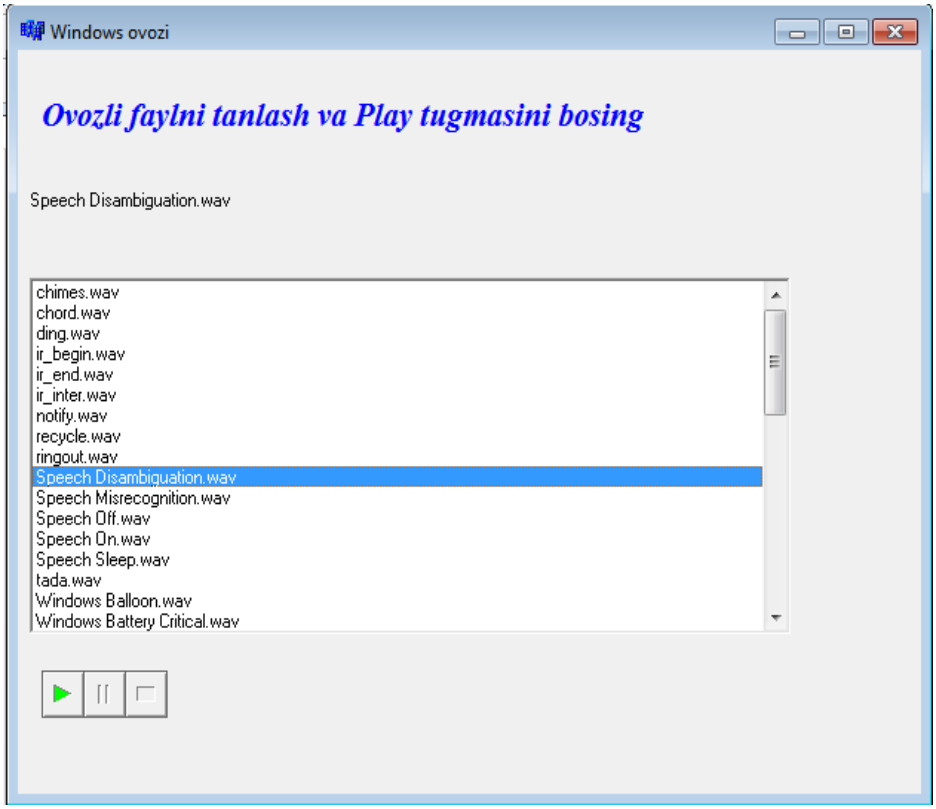
```
#include <vcl.h>

#pragma hdrstop

#include "animate2.h"

#include<string.h>

//---------------------------------------------------------------------------

#pragma package(smart_init)

#pragma resource "*.dfm"
```

```
TForm1 *Form1;
String SoundPath;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{   char *wd;  // êàòàëîã Windows
wd = (char*)AllocMem(MAX_PATH);
GetWindowsDirectory(wd,MAX_PATH);
SoundPath = wd;
SoundPath = SoundPath + "\\Media\\";
TSearchRec sr;
if (FindFirst( SoundPath + "*.wav", faAnyFile, sr) == 0)
{   ListBox1->Items->Add(sr.Name); // äîáàâèì èìÿ ôàéëà â ñïèñîê
while (FindNext(sr) == 0)
ListBox1->Items->Add(sr.Name);
if (FindFirst( SoundPath + "*.mid", faAnyFile, sr) == 0)
```

```
ListBox1->Items->Add(sr.Name); // äîáàâèì èìÿ ôàéëà â ñïèñîê

while (FindNext(sr) == 0)

ListBox1->Items->Add(sr.Name);  }

if (FindFirst( SoundPath + "*.rmi", faAnyFile, sr) ==0)

ListBox1->Items->Add(sr.Name); // äîáàâèì èìÿ ôàéëà â ñïèñîê

while (FindNext(sr) == 0)

ListBox1->Items->Add(sr.Name);

if ( ListBox1->Items->Count != 0)

{  Label2->Caption = ListBox1->Items->Strings [1] ;

MediaPlayer1->FileName = SoundPath + ListBox1->Items->Strings [1] ;

MediaPlayer1->Open ( ) ;

MediaPlayer1->Play() ;} }

//---------------------------------------------------------------------------

void __fastcall TForm1::ListBox1Click(TObject *Sender)

{ Label2->Caption = ListBox1->Items->Strings [ListBox1->ItemIndex];

MediaPlayer1->FileName = SoundPath + Label2->Caption;

MediaPlayer1->Open() ;

MediaPlayer1->Play() ;}
```

Dastur quyidagicha ishlaydi: Dastur ishga tushirilishi bilan *OnCreate* hodisasi Windowsning Media katalogida joylashgan ovozli fayllar ro'yxatini tuzadi. Turli operatsion sistamalarda bu katalog turlicha nomlanga bo'lishi mumkin.

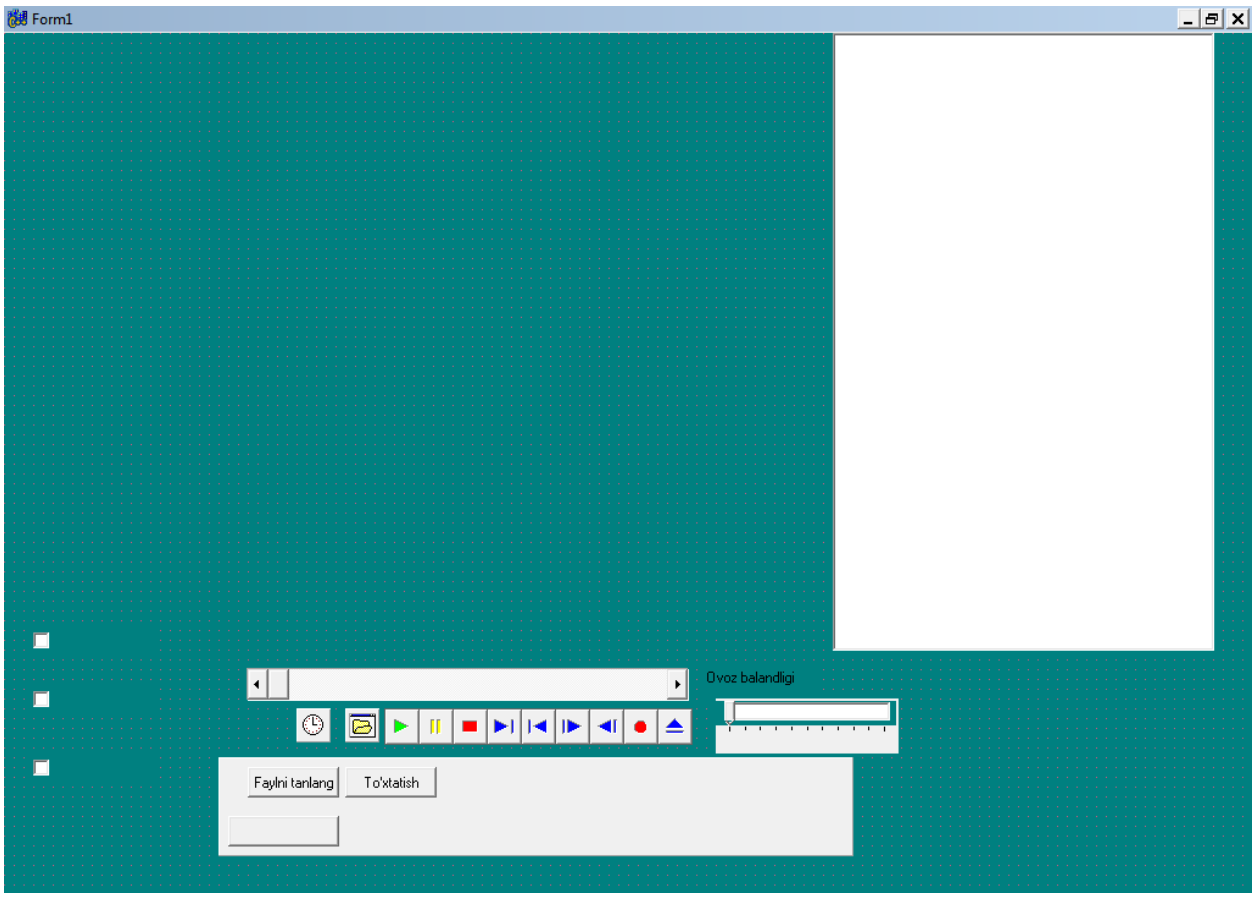
Shuning uchun uning nomini olish maqsadida *GetWindows Directory* uchun API funksiyasidan foydalaniladi. *FineFirst* va *FindNext* funksiyalaridan foydalanib ovozli fayllarning royxati tashkil etiladi. Bu funksiyalar bajarilgach tashkil etilgan ovozli fayllarning birinchisi ustida marker joylashtiriladi. Quyidagi misol ovozli faylni kompakt diskdan o'qish jarayonini amalga oshirishga mo'ljallangan. 6- rasmda dasturning formadagi ko'rinishi keltirilgan. Formada *MediaPlayer* komponentasi joylashtirilgan bo'lib *Visible* xossasiga *false* qiymatlangan. Shuning uchun u dastur ishga tushganda ko'rinmaydi



3.3. Rasm.Ovozli fayllar bilan ishlovchi dastur natijasi

3.3. Video fayllar bilan ishlash

MediaPlayer komponentasidan foydalaninb videoroliklarni ko'rish uchun mo'ljallangan forma ko'rinishi quyidagicha:



3.4. Rasm. Video fayllar bilan ishlovchi dastur formasining ko’rinishi

Quyida videorolikni ko’rishga mo’ljallangan dastur matnini keltiramiz:

```
#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart_init)

#pragma resource "*.dfm"

TForm1 *Form1;

#define MINUTE(ms) ((ms/1000)/60)
```

```
#define SECOND(ms) ((ms/1000)%60)

Graphics::TBitmap *fcmPlay,*kmPlay, *bmPlay; // Play

Graphics::TBitmap *fcmPause,*bmPause; // Pause

//---------------------------------------------------------------------------

__fastcall TForm1::TForm1(TComponent* Owner)

      : TForm(Owner)

{

bmPlay = new Graphics::TBitmap();

bmPause = new Graphics::TBitmap();

SpeedButton2->Glyph->Assign(bmPlay);

MediaPlayer1->Display = Form1;}

void _fastcall GetFrameSize (AnsiString f, int *w, int *h)

{  if ( f.Pos(".avi") == 0 )

{  *w = 352;

*h = 240;

struct {

char RIFF[4];

 long int nu_1[5];

char AVIH[4];

long int nu_2[9];
```

```
long int w;

long int h;

} header;

TFileStream *fs;

fs = new TFileStream(f,fmOpenRead) ;

fs->Read(&header, sizeof(header));

*w = header.w;

*h = header.h;

delete  fs; }  }

//---------------------------------------------------------------------------

void __fastcall TForm1::SpeedButton1Click(TObject *Sender)

{  int fw, fh;

int top,left;

int sw, sh;

int mw, mh;

float kw, kh;

float k;

OpenDialog1->Title = "Tanlash";

OpenDialog1->InitialDir = "";

OpenDialog1->Filter =
```

```
"Все файлы|*.avi;*.mpg;*.mpegI"

"AVI|*.avi|MPG|*.mpg|MGEG|*.mpeg";

if ( ! OpenDialog1->Execute() )

return;

if ( MediaPlayer1->FileName==OpenDialog1->FileName )

return;

GetFrameSize(OpenDialog1->FileName,&fw, &fh);

mw = Form1->ClientWidth;

mh = Form1->Panel1->Top-10;

if ( fw < mw )

kw = 1;

else kw = (float) mw / fw;

if ( fh < mh )

kh = 1; else kh = (float) mh / fh;

if ( kw < kh )

k = kw;

else k = kh;

sw = fw * k;

sh=fh*k;

left = (Form1->ClientWidth - sw) / 2;
```

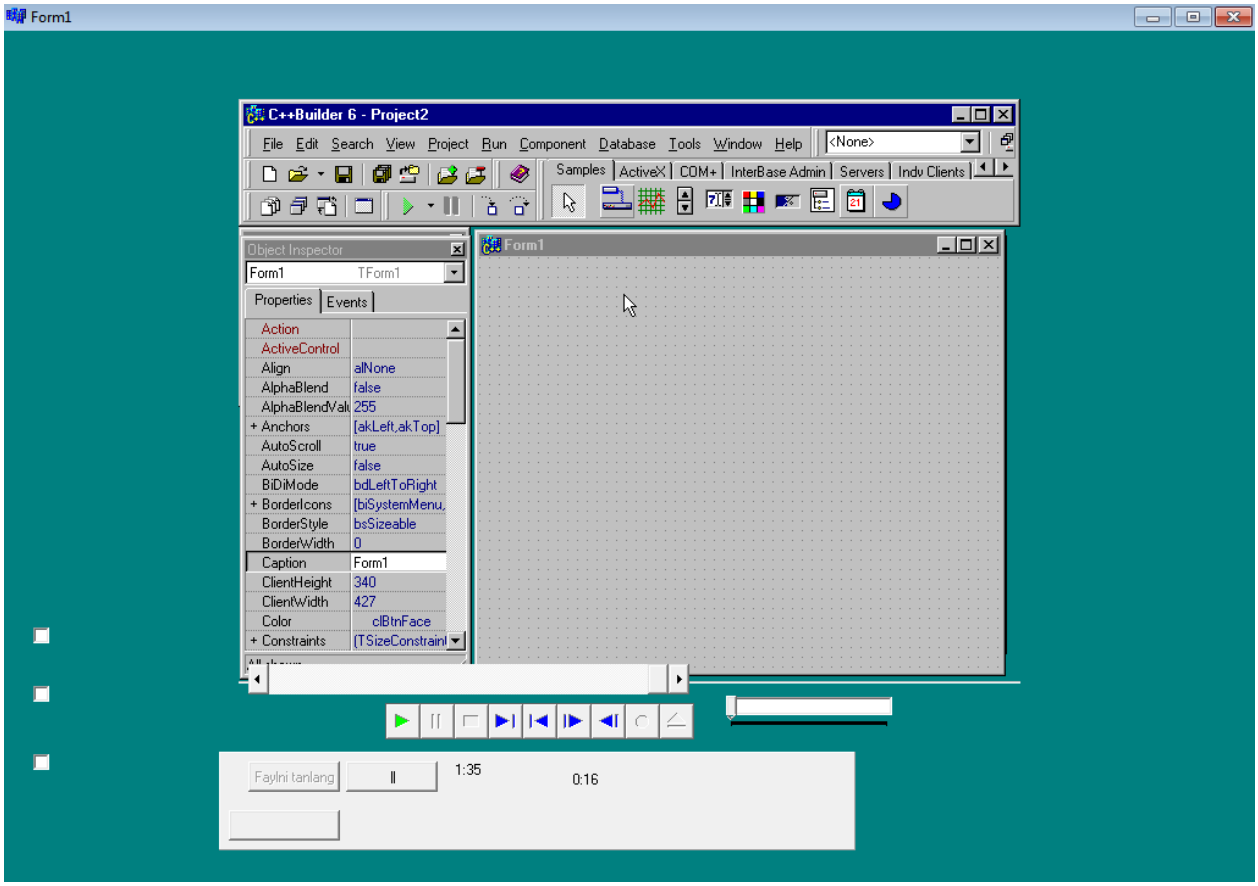
```
top = (Panel1->Top - sh) / 2;

MediaPlayer1->FileName = OpenDialog1->FileName;

MediaPlayer1->Open();

MediaPlayer1->DisplayRect = Rect(left-400,top+20,sw-400,sh-110);

Form1->Canvas->FillRect(Rect(0,0,ClientWidth,Panel1->Top));

SpeedButton2->Enabled = true;

MediaPlayer1->TimeFormat = tfMilliseconds;

int ms = MediaPlayer1->Length;

AnsiString st = IntToStr(SECOND(ms));

if ( st.Length() == 1)

st = "0" + st;

st = IntToStr(MINUTE(ms)) + ":" + st;

Label1->Caption = st;

Label3->Caption = "0:00";

SpeedButton2->Glyph->Assign(bmPause);

SpeedButton2->Hint = "Pause";

SpeedButton2->Tag = 1;

SpeedButton1->Enabled = False;

MediaPlayer1->Play();

Timer1->Enabled = true ;}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{  if (SpeedButton2->Tag == 0)
{  SpeedButton2->Glyph->Assign(bmPause);
SpeedButton2->Hint = "Pause";
SpeedButton2->Tag = 1;
SpeedButton1->Enabled = False; // êíîïêà Eject
MediaPlayer1->Play();
Timer1->Enabled = true;}
else {
MediaPlayer1->Stop();
SpeedButton2->Glyph->Assign(bmPlay);
SpeedButton2->Hint = "Play";
SpeedButton2->Tag = 0;
SpeedButton1->Enabled = True;
Timer1->Enabled = false; }  }
//---------------------------------------------------------------------------
void __fastcall TForm1::MediaPlayer1Notify(TObject *Sender)
{  if ( ( MediaPlayer1->Mode == mpStopped ) &&
( SpeedButton2->Tag == 1))
```

```
{  Timer1->Enabled= false;
SpeedButton2->Glyph->Assign(bmPlay);
SpeedButton2->Hint = "Play";
SpeedButton2->Tag = 0;
SpeedButton1->Enabled = True; } }
//---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{ if ( MediaPlayer1->Mode == mpStopped )
{ MediaPlayer1->Position = 1;
MediaPlayer1->Position = 0;
}  }
//---------------------------------------------------------------------------
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{  MediaPlayer1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{  MediaPlayer1->TimeFormat = tfMilliseconds;
int ms = MediaPlayer1->Position;
AnsiString st = IntToStr(SECOND(ms));
```

```
if ( st.Length() == 1)

st = "0" + st;

st = IntToStr(MINUTE(ms)) + ":" + st;

Label3->Caption = st;

}
```

Dastur ishga tushirilgach, kerakli video faylni tanlab, uning namoyishini boshlash mumkin:



3.5. Rasm. Video fayllar bilan ishlovchi dastur natijasi

Bu kabi dasturiy vositalar yordamida audio va video fayllar bilan ishlovchi dasturlarning qo'llanmalilik darajasini oshirish, foydalanuvchu istagidan kelib chiqib, o'zgartirishlar kiritish mumkin.

XULOSA

Ushbu bitiruv malakaviy ishida qo'yilgan "C++ tilining multimedia imkoniyatlari" mavzusi ostida qo'yilgan maqsad va vazifalarni bajarish davomida quyidagi natijalarga erishildi:

1) C++ tilining grafika va multimedia bilan ishlovchi komponentalari batafsil o'rganib chiqildi va ish tarkibiga kiritildi;
2) Ishda grafik asos va asosiy grafik primitivlar haqida umumiy tushunchalar keltirildi;
3) Tasvirlarni tashqi fayldan o'qish va bitli tasvirlar bian ishlash to'g'risida ma'lumotlar keltirildi va amaliy masalalar yordamida yoritildi;
4) Multiplikatsiyalar hosil qilish jarayoni o'rganib chiqildi va amaliy dasturiy vositalar yordamida yoritildi;
5) C++ tilida ovozli va ovozsiz animatsiyalar bilan ishlovchi Animate va MediaPlayer komponentalarining xususiyatlari va imkoniyatlari o'rganib chiqildi va ishda keltirildi;
6) Koordinatalar sistemasini boshqaruvchi Chart koordinatasi imkoniyatlari o'rganildi va bu komponenta yordamida bitta koordinatalar sistemasida bir nechta funksiyalarning grafiklarini quruvchi dasturiy vosita yaratildi;
7) MediaPlayer komponentasidan foydalangan hold ovozli fayllarni ijro etuvchi dasturiy vosita yaratildi va ish tarkibiga kiritildi;
8) Video fayllarni ijro etuvchi dastur yaratildi va ish tarkibiga kiritildi.

Ushbu bitiruv ishida tuzilgan animatsiyani, ovozli ijroni, video tasvirni boshqarishga mo'ljallangan dasturiy vositalardan foydalanib, multimdiali elektron o'quv qo'llanmalar yaratish va audio va videopleyer dasturlarining qo'llanmalilik darajasini yuksaltirish kabi ishlarni amalga oshirish mumkin.

FOYDALANILGAN ADABIYOTLAR VA INTERNET RESURSLARI

1. Н.Культин. Самоучитель C++ Builder СПб.: БХВ-Петербург, 2004. -320 с.:
2. Н.Культин. C++ Builder в задачах и примерах. Петербург, 2005. — 336 с:
3. Х.М.Дейтел, П.Дж.Дейтел «Как программировать на C++», 5-издание, М-2008 г. 1454 стр.
4. Г.Шилдт –“Полный справочник по C++” – М-2006., 801 стр.
5. Sh. F. Madraximov “C++. Obyektga yo’naltirilgan dasturlash”, Toshkent, Mumtoz so’z, 2016 yil. 176 s.
6. Р.Седжвик – “Фундаментальные алгоритмы на C++” – М 2001., 687 стр.
7. Ш.Ф.Мадрахимов, С.М.Гайназаров “C++ тилида программалаш асослари” – Тошкент-2009 й., 196
8. М.Э.Абрамян “Электронный задачник по программированию” Ростов - на - Дону 2005 г. 182 стр.
9. Н.Н.Непейвода- Стили и методы программирования. Интернет университет информационных технологий. INTUIT.ru, 2005 г., 320 стр.
10. O.M.Shukurov, E.A.Eshboyev, B.H.Shovaliyev – “Delphi va C++ algoritmik tillarida dasturlash” – Qarshi-2012 y., 228 s.
11. E.A.Eshboyev, F.Yu.Shodiyev, F.G.Qlicheva – “C++ tilida dasturlash” – Toshkent-2014 yil. 206 s.
12. www.cplusplus.com
13. www.acm.timus.ru
14. www.codeforces.ru
15. www.cyberforum.ru

16.www.cybern.ru

17.www.delphisources.ru

18.www.ziyonet.uz

19. www.dastur.uz