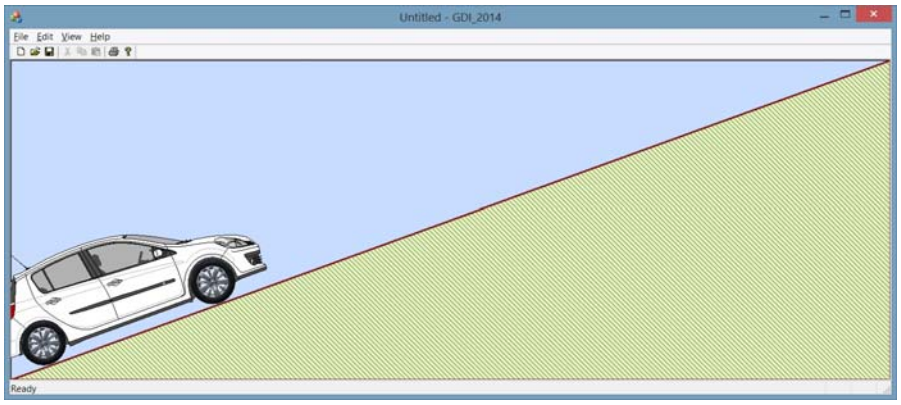
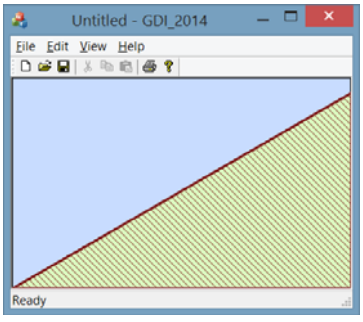
GDI-2014

I kolokvijum

rešenje



Napisati funkciju **DrawGround(CDC* pDC, float angle)**, koja iscrtava podlogu po kojoj se kreće automobil. Podloga je uzdignuta sleva udesno pod uglom **angle** zadatom u stepenima (počinje u donjem levom uglu prozora). Okvir je iscrtan olovkom debljine 3 braon boje, a ispuna šrafurom braon-svetlo-zelene boje. Ostatak ekrana obojen je svetlo plavo. [10 poena]




```
void CGDI_2014View::DrawGround(CDC* pDC, float angle)
{
CRect rect;
GetClientRect(&rect);
CPen pen(PS_SOLID, 3, RGB(128, 32, 32));
CPen* pOldPen = pDC->SelectObject(&pen);
CBrush brush(HS_FDIAGONAL, RGB(128, 32, 32));
CBrush* pOldBrush = pDC->SelectObject(&brush);
POINT pts[3];
pts[0].x = rect.left; pts[0].y = rect.bottom;
pts[1].x = rect.right; pts[1].y = rect.bottom;
pts[2].x = rect.right; pts[2].y = rect.bottom - rect.Width()*tan(angle*toRad);
COLORREF oldBkColor = pDC->SetBkColor(RGB(220, 255, 192));
pDC->Polygon(pts, 3);
pDC->SetBkColor(oldBkColor);
pDC->SelectObject(pOldPen);
pDC->SelectObject(pOldBrush);
}
```

U konstruktoru **View** klase ili u funkciji **OnInitialUpdate** učitati sliku **Wheel.png** i metafajl **clio.emf**, i obrisati ih u destruktoru. [10 poena]

```
CGDI_2014View::CGDI_2014View()
{
	m_hPos = 0.0f;
	m_angle = 10.0f;

	m_pTocak = new DImage();
	m_pTocak->Load(CString("Wheel.png"));
	m_met = GetEnhMetaFile(CString("clio.emf"));

}

CGDI_2014View::~CGDI_2014View()
{
	delete m_pTocak;
	DeleteEnhMetaFile(m_met);
}
```

Napisati funkciju **Translate(CDC* pDC, float dX, float dY, bool rightMultiply)**, koja modifikuje trenutnu svetsku transformaciju, tako da vrši translaciju za date vrednosti po X i Y-osi. Poslednji parametar definiše da li se množenje matrice obavlja sa desne strane (true) ili leve strane (false). [5 poena]

```
void CGDI_2014View::Translate(CDC* pDC, float dX, float dY, bool
rightMultiply)
{
XFORM trans;
trans.eM11 = 1.0;
trans.eM12 = 0.0;
trans.eM21 = 0.0;
trans.eM22 = 1.0;
trans.eDx = dX;
trans.eDy = dY;
if (rightMultiply)
	pDC->ModifyWorldTransform(&trans, MWT_RIGHTMULTIPLY);
else
	pDC->ModifyWorldTransform(&trans, MWT_LEFTMULTIPLY);
}
```

Napisati funkciju **Rotate(CDC* pDC, float angle, bool rightMultiply)**, koja modifikuje trenutnu svetsku transformaciju, tako da vrši rotaciju za ugao **angle** zadat u stepenima. [5 poena]

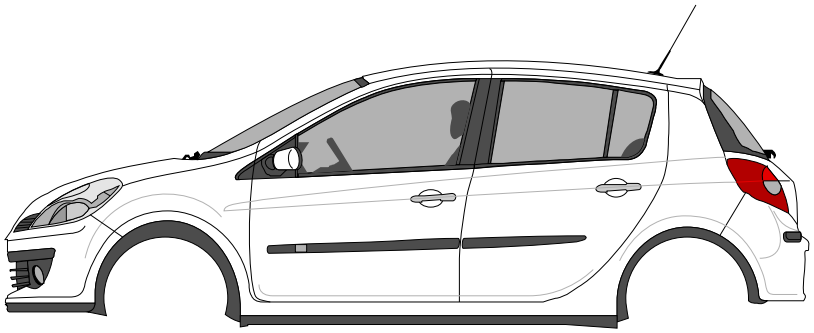
```
void CGDI_2014View::Rotate(CDC* pDC, float angle, bool rightMultiply)
{
XFORM trans;
trans.eM11 = cos(angle*toRad);
trans.eM12 = sin(angle*toRad);
trans.eM21 =-sin(angle*toRad);
trans.eM22 = cos(angle*toRad);
trans.eDx = 0.0;
trans.eDy = 0.0;
if (rightMultiply)
        pDC->ModifyWorldTransform(&trans, MWT_RIGHTMULTIPLY);
else
        pDC->ModifyWorldTransform(&trans, MWT_LEFTMULTIPLY);
}
```

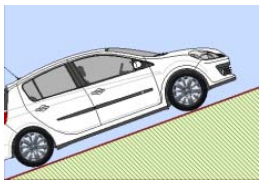
Napisati funkciju **Scale(CDC* pDC, float sX, float sY, bool rightMultiply)**, koja modifikuje trenutnu svetsku transformaciju, tako da vrši skaliranje za date vrednosti po X i Y-osi. [5 poena]

```
void CGDI_2014View::Scale(CDC* pDC, float sX, float sY, bool rightMultiply)
{
XFORM trans;
trans.eM11 = sX;
trans.eM12 = 0.0;
trans.eM21 = 0.0;
trans.eM22 = sY;
trans.eDx = 0.0;
trans.eDy = 0.0;
if (rightMultiply)
        pDC->ModifyWorldTransform(&trans, MWT_RIGHTMULTIPLY);
else
        pDC->ModifyWorldTransform(&trans, MWT_LEFTMULTIPLY);
}
```

Napisati funkciju **DrawCar(CDC* pDC, int x, int y, int w, int h)** koja prikazuje metafajl **clio.emf** sa centrom u tački **(x,y)**, širine **w** i visine **h**. Pri iscrtavanju automobil je „okrenut“ na desnu stranu (vidi sliku). [10 poena]

```
void CGDI_2014View::DrawCar(CDC* pDC, int x, int y, int w, int h, float
angle)
{
pDC->PlayMetaFile(m_met, CRect(x + w / 2, y - h / 2, x - w / 2, y + h / 2));
}
```



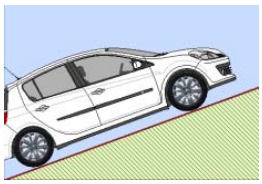


Napisati funkciju **DrawWheel(CDC* pDC, int x, int y, int r, float angle)** koja iscrtava bitmapu **Wheel.png** sa centrom u tački **(x,y)**, poluprečnika **r** i zarotiran za ugao **angle** u stepenima. Za transformaciju koristiti prethodno definisane funkcije. Točak se u slici Wheel.png nalazi na poziciji definisanom okvirnim pravougaonikom sa gornjim levim uglom na poziciji (52,15) i širine/visine jednake 376 piksela. Iscrtavanje ograničiti samo na dati okvirni pravougaonik. Za iscrtavanje koristiti funkciju **DrawTransparent** klase **DImage**, koja iscrtava datu bitmapu uz transparenciju zadate boje. [15 poena]

```
void CGDI_2014View::DrawWheel(CDC* pDC, int x, int y, int r, float angle)
{
XFORM transOld;
pDC->GetWorldTransform(&transOld);
Translate(pDC, x, y, false); // MWT_LEFTMULTIPLY
Rotate(pDC, angle, false); // MWT_LEFTMULTIPLY

int dx = 52, dy = 15, w = 376;
CRect srcRc(dx, dy, dx + w, dy + w);
CRect dstRc(-r, -r, r, r);
m_pTocak->DrawTransparent(pDC, srcRc, dstRc, RGB(255,255,255));
pDC->SetWorldTransform(&transOld);
}
```

Korišćenjem prethodnih funkcija iscrtati telo automobila širine 450 jedinica i visine 2.5 puta manje od širine, i dva točka poluprečnika 38 na odgovarajućim pozicijama (središta oba točka su spuštena za 70 jedinica u odnosu na centar tela automobila, levi je pomeren -155, a desni +135 po X-osi). Čitav automobil postaviti na početku strme ravni, tako da točkovi dodiruju strmu ravan (vidi sliku). Prilikom kretanja automobila, točkovi se rotiraju u skladu sa pređenim putem. [20 poena]

```
void CGDI_2014View::OnDraw(CDC* pDC)
{
CRect rect;
GetClientRect(&rect);

CDC* pMemDC = new CDC();
pMemDC->CreateCompatibleDC(pDC);
CBitmap bmp;
bmp.CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());
pMemDC->SelectObject(&bmp);
CBrush brush(RGB(200, 220, 255));
CBrush* pOldBrush = pMemDC->SelectObject(&brush);
pMemDC->Rectangle(0, 0, rect.Width(), rect.Height());
pMemDC->SelectObject(pOldBrush);

XFORM transOld;
int gm = pMemDC->SetGraphicsMode(GM_ADVANCED);
pMemDC->GetWorldTransform(&transOld);
float angle = m_angle;
DrawGround(pMemDC, angle);
```



```
int carWidth = 450;
int carHeight = carWidth/2.5;
int x = 0 + m_hPos, y = 0, r = 38;
float alpha = toDeg * m_hPos / (2 * 3.141592653589793238);

pMemDC->GetWorldTransform(&transOld);

Translate(pMemDC, carWidth / 2.0, -carHeight / 2.0 - r / 2.0, true);
Rotate(pMemDC, -angle, true); // MWT_RIGHTMULTIPLY
Translate(pMemDC, 0.0, rect.Height(), true);

DrawCar(pMemDC, x, y, carWidth, carWidth / 2.5, 0.0f); // Kola

DrawWheel(pMemDC, x - 155, y + 70, r, alpha); // tockovi
DrawWheel(pMemDC, x + 135, y + 70, r, alpha);

pMemDC->SetWorldTransform(&transOld);
pMemDC->SetGraphicsMode(gm);

pDC->BitBlt(0, 0, rect.Width(), rect.Height(), pMemDC, 0, 0, SRCCOPY);
pMemDC->DeleteDC();
delete pMemDC;
}
```

Pritiskom na taster → na tastaturi omogućiti pomeranje automobila udesno, uz strmu ravan, za 10 jedinca. Pritiskom na taster ← omogućiti kretanje automobila unazad. Pritiskom na taster ↑ povećati nagib strme ravni za 10°, a pritiskom na taster ↓ smanjiti nagib za 10°. Pri svakoj promeni nagiba automobil vratiti na početak strme ravni. Nagib ograničiti na opseg vrednosti [-10°, +80°]. [10 poena]

```
void
CGDI_2014View::OnKeyDown(UINT
nChar, UINT nRepCnt, UINT nFlags)
{
if (nChar == VK_RIGHT)
{
        m_hPos += 10.0;
        Invalidate();
}
if (nChar == VK_LEFT)
{
        m_hPos -= 10.0;
        if (m_hPos < 0.0)
                m_hPos = 0.0;
        Invalidate();
}
if (nChar == VK_UP)
{
        m_hPos = 0.0;
        m_angle += 10.0f;
        if (m_angle > 80.0f)
                m_angle = 80.0f;
        Invalidate();
}
if (nChar == VK_DOWN)
{
        m_hPos = 0.0;
        m_angle -= 10.0f;
        if (m_angle < -10.0f)
                m_angle = -10.0f;
        Invalidate();
}

CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

Eliminisati *flicker* korišćenjem memorijskog DC-ja i sprečavanjem brisanja prozora. [10 poena]

```
BOOL CGDI_2014View::OnEraseBkgnd(CDC* pDC)
{
        return TRUE;
}

void CGDI_2014View::OnDraw(CDC* pDC)
{
CRect rect;
GetClientRect(&rect);

CDC* pMemDC = new CDC();
pMemDC->CreateCompatibleDC(pDC);
CBitmap bmp;
bmp.CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());
pMemDC->SelectObject(&bmp);
//...
pDC->BitBlt(0, 0, rect.Width(), rect.Height(), pMemDC, 0, 0, SRCCOPY);
pMemDC->DeleteDC();
delete pMemDC;
}
```