Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

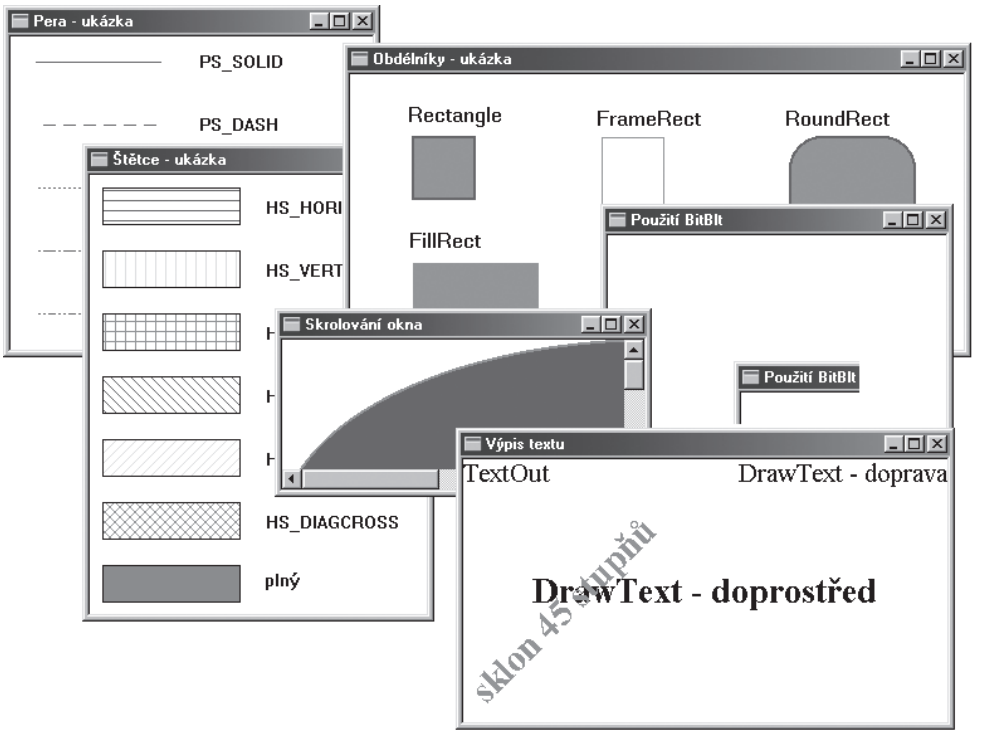
To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

4

KRESLENÍ DO OKNA



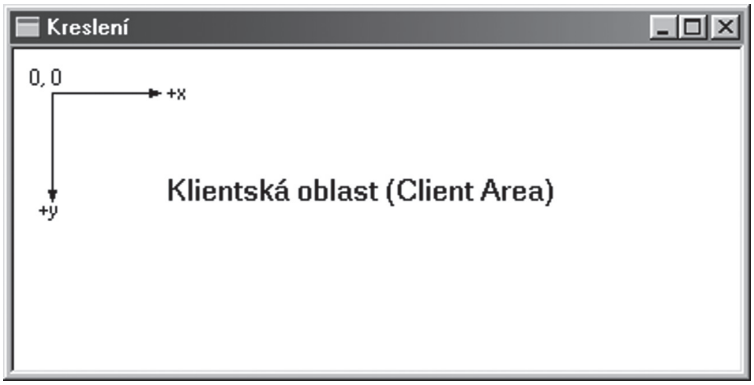
V této kapitole se seznámíme s filozofií kreslicích operací z pohledu Windows a MFC.

4.1 ZÁKLADNÍ INFORMACE

Aplikace Windows realizují kreslicí operace do okna zpracováním zprávy **WM_PAINT**. Této zprávě odpovídá v modelu MFC handler **OnPaint**.

Kreslicí plocha (okno, ale i třeba papír vložený v tiskárně) může určovat souřadnice různým způsobem, mluvíme o tzv. *mapování souřadnic*. Tento systém dovoluje přizpůsobit rozměry použitým jednotkám (pixely, centimetry nebo palce) či definovat počátek souřadného systému. V této kapitole budeme uvažovat výchozí souřadný systém označovaný jako **MM_TEXT**. Jak název napovídá, je tento mapovací režim důležitý pro výpis textu.

V režimu **MM_TEXT** je levý horní roh klientské oblasti okna brán jako souřadnice 0, 0. Souřadnice x narůstá směrem zleva doprava, souřadnice y narůstá shora dolů. Přesně tak, jak se má vypisovat text. Každý bod v tomto prostoru odpovídá jednomu pixelu obrazovky. Situace je zřejmá z *obr. 4.1*.



*Obr. 4.1 Mapování souřadnic v režimu **MM_TEXT***

4.2 WINDOWS GDI

Operační systém definuje jako svou součást rozhraní **GDI** (Graphic Device Interface). Toto rozhraní zajišťuje jednotný přístup k libovolné kreslicí ploše (například okno nebo stránka papíru v tiskárně). To velmi zjednodušuje programování, protože jsme odstíněni od závislosti na hardwaru.

Model GDI vychází z tzv. *kontextu zařízení* (**DC** – device context). Kontext zařízení v sobě obsahuje veškerá nastavení nutná pro zajištění kreslicích operací (například sílu pera, jeho barvu a styl atd.).

MFC zapouzdřuje kontext zařízení pomocí bázové třídy **CDC**. Z této třídy se pak odvozují další třídy, které se specializují na konkrétní operace. Seznam je uveden formou *tab. 4.1*.

*Tab. 4.1 Následnické třídy k **CDC***

Třída	Popis
CPaintDC	kreslí do klientské oblasti okna (pouze v handleru **OnPaint**),
CClientDC	kreslí do klientské oblasti okna (pouze mimo handler **OnPaint**),
CWindowDC	kreslí do klientské i neklientské oblasti okna (například rámeček a titulek),
CMetaFileDC	kreslí do metasouboru (ukládají se grafické příkazy, kreslení se fyzicky neprovádí).

4.2.1 Třída CPaintDC

V popředí našeho zájmu bude třída **CPaintDC**. Ta umožňuje kreslení v rámci odezvy na zprávu **WM_PAINT** (kód se tedy zapisuje do handleru **OnPaint**).

Obvyklé je vytvořit instanci třídy **CPaintDC** jako automatickou (třídy auto, alokovanou na zásobníku) právě v handleru **OnPaint** (tedy ne operátorem **new**). Konstruktor této třídy totiž ve své režii volá funkci Windows API nazvanou **BeginPaint** (ta zajistí získání prostředků pro kreslení). Na konci handleru se automatické objekty destruují (je totiž nutno uvolnit zásobník) a tak se volá destruktor, ten zase zajistí vyvolání funkce Windows API nazvané **EndPaint**, tím kreslicí operace končí.

4.2.2 Třída CClientDC

V některých příkladech potřebujeme provést kreslení okamžitě, tedy ne až po příjmu zprávy **WM_PAINT**. Pak je místo instance třídy **CPaintDC** nutno vytvořit instanci třídy **CClientDC**. Příklad je uveden v kapitole 4.9.1.

4.2.3 Třída CWindowDC

V ojedinělých případech potřebujeme kreslit nejen do klientské oblasti, ale také do neklientské oblasti (rámeček okna, titulkový pruh apod.). Pak použijeme instanci třídy **CWindowDC**. Příklady jsou uvedeny v kapitolách 4.9.9 a 4.9.10.

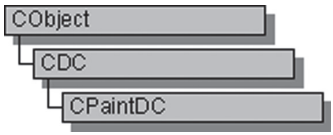
4.2.4 Třída CMetaFileDC

Pokud použijeme instanci třídy **CMetaFileDC**, nebudou se kreslicí příkazy provádět, ale uloží se do tzv. *metasouboru*. Metasoubor obsahuje kódy grafických operací a jejich parametry. Tento soubor lze *přehrát* a tak grafické příkazy provést. Metasoubory uložené na disk mají příponu buď **WMF** (pochází ze 16bitových Windows) nebo **EMF** (rozšířený metasoubor ze 32bitových Windows).

4.3 POPIS TŘÍDY CPaintDC

Třída **CPaintDC** je určena pro kreslení do klientské oblasti okna realizované v handleru **OnPaint**.

Samotná třída disponuje pouze dvěma novými atributy, zbytek je přejat z báze **CDC**. V našem popisu se zaměříme pouze na přehled atributů a metod, protože jejich počet je poměrně vysoký.



*Obr. 4.2 Hierarchie třídy **CPaintDC***

4.3.1 Atributy

- **PAINTSTRUCT m_ps** – informace použitelné pro kreslení, z celé struktury jsou patrně nejdůležitější tyto položky:
  - **HDC hdc** – handle kontextu zařízení (lze použít pro přímé volání funkcí Windows GDI),
  - **BOOL fErase** – určuje, zda má být pozadí překresleno. Pokud je **fErase** různé od 0, pozadí by mělo být překresleno,
  - **RECT rcPaint** – určuje souřadnice obdélníka, který by měl být překreslen, aby došlo k obnovení obrazu (použitelné, pokud dokážeme překreslit jen část obrázku, která byla poškozena),
- **HWND m_hWnd** – handle okna, do kterého se kreslí,
- **HDC m_hDC** – handle kontextu zařízení (lze použít pro přímé volání funkcí Windows GDI).

4.3.2 Metody

- **CPaintDC**(CWnd* pWnd) – konstruktor, **pWnd** uvádí instanci třídy odvozené z **CWnd** (například následníka **CFrameWnd**), do jehož okna se bude kreslit. Protože je instance třídy **CPaintDC** obvykle vytvořena v handleru **OnPaint** dané třídy okna, bude ve většině případů platit **pWnd = this**,
- **CBitmap *GetCurrentBitmap**( ) – vrátí ukazatel na instanci třídy **CBitmap**, která je vybrána jako aktivní,
- **CBrush *GetCurrentBrush**( ) – vrátí ukazatel na instanci třídy **CBrush**, která je vybrána jako aktivní. **CBrush** zapouzdřuje operace nad štětcem (vyplňování ploch),
- **CPen *GetCurrentPen**( ) – vrátí ukazatel na instanci třídy **CPen**, která je vybrána jako aktivní. **CPen** zapouzdřuje operace nad perem (kreslení obrysů),

- **CPalette *GetCurrentPalette**( ) – vrátí ukazatel na instanci třídy **CPalette**, která je vybrána jako aktivní. **CPalette** zapouzdřuje operace nad paletou (potřebné například pro paletovou animaci),
- **CFont *GetCurrentFont**( ) – vrátí ukazatel na instanci třídy **CFont**, která je vybrána jako aktivní. **CFont** zapouzdřuje operace nad písmem (výpis textu),
- **int SaveDC**( ) – uloží stav kontextu zařízení do zásobníku a vrátí odkaz na takto uložené parametry v podobě jeho pořadového čísla (při chybě vrátí 0),
- **BOOL RestoreDC**(int nSavedDC) – obnoví stav kontextu zařízení ze zásobníku podle indexu udaného parametrem **nSavedDC**. Je-li **nSavedDC = –1**, obnoví se posledně uložený stav. Při chybě vrátí **FALSE**,
- **int GetDeviceCaps**(int nIndex) – zjistí schopnosti zařízení. Parametr **nIndex** udává charakteristiku, kterou chceme zjistit (popis je vynechán, najdete jej v nápovědě),
- **CPoint GetBrushOrg** – zjistí počáteční pozici, od které se kreslí štětcem jako instanci třídy **CPoint**. Výchozí hodnota je 0, 0 (změnu lze zajistit pomocí metody **SetBrushOrg**),
- **CPoint SetBrushOrg**(int x, int y) – nastaví počáteční pozici štětce podle **x**, **y**,
- **CPoint SetBrushOrg**(POINT point) – nastaví počáteční pozici štětce podle souřadnic předaných strukturou **POINT**,
- **SelectObject** (formy viz *tab. 4.2*) – nastaví aktuální objekt GDI, takto se například vybere aktuální pero založené instancí třídy **CPen**,

*Tab. 4.2 Možné formy metody **CPaintDC::SelectObject***

Forma	Stručný popis
CPen* SelectObject(CPen* pPen)	vybere aktuální pero, vrátí ukazatel na dříve aktivní pero,
CBrush* SelectObject(CBrush* pBrush)	vybere aktuální štětec, vrátí ukazatel na dříve aktivní štětec,
CFont* SelectObject(CFont* pFont)	vybere aktuální font, vrátí ukazatel na dříve aktivní font,
CBitmap* SelectObject(CBitmap* pBitmap)	vybere aktuální bitmapu, vrátí ukazatel na dříve aktivní bitmapu,
int SelectObject(CRgn* pRgn)	vybere omezující oblast kreslení, indikuje typ vybrané oblasti.

- **CGdiObject* SelectStockObject**(int nIndex) – nastaví jako aktuální objekt některé z per, štětců, fontů předvytvořených operačním systémem. Hodnota **nIndex** určuje objekt GDI, který chceme vybrat. Funkce vrací ukazatel na instanci **CGdiObject** (z této třídy jsou odvozeny třídy **CPen**, **CBrush** apod., viz *obr. 4.6*), která byla dříve aktivní,

*Tab. 4.3 Možné hodnoty **nIndex** pro metodu **CPaintDC::SelectStockObject***

Hodnota	Význam
BLACK_BRUSH	černý štětec
GRAY_BRUSH	šedý štětec
LTGRAY_BRUSH	světle šedý štětec
DKGRAY_BRUSH	tmavě šedý štětec
WHITE_BRUSH	bílý štětec
NULL_BRUSH	prázdný štětec (nekreslí)
HOLLOW_BRUSH	stejné jako **NULL_BRUSH**
BLACK_PEN	černé pero
WHITE_PEN	bílé pero
NULL_PEN	prázdné pero (nekreslí)
ANSI_FIXED_FONT	ANSI systémový font s pevnou šíří znaků
ANSI_VAR_FONT	ANSI systémový font s proměnlivou šíří znaků
SYSTEM_FONT	systémový font s proměnlivou šíří znaků (je používán pro kreslení menu, dialogů a dalšího textu)
SYSTEM_FIXED_FONT	systémový font s pevnou šíří znaků, je zaveden pro kompatibilitu s Windows 3.0
DEFAULT_PALETTE	výchozí paleta barev obsahující 20 statických barev systémové palety

- **COLORREF GetBkColor**( ) – vrátí barvu používanou pro výplň pozadí pokud je zvolen režim pozadí **OPAQUE**,
- **COLORREF SetBkColor**(COLORREF crColor) – nastaví barvu **crColor** pro výplň pozadí v režimu pozadí **OPAQUE**, vrací dříve vybranou barvu,
- **int GetBkMode**( ) – vrátí aktuální režim pozadí, viz *tab. 4.4*,
- **int SetBkMode**(int nBkMode) – nastaví režim pozadí podle parametru **nBkMode** (viz *tab. 4.4*), vrátí předchozí režim pozadí,

Tab. 4.4 Režimy pozadí

Hodnota	Význam
OPAQUE	pozadí je vyplněno aktuální barvou pozadí před tím, co je vypsán text, použito pero nebo štětec. Výchozí režim.
TRANSPARENT	pozadí se před kreslicí operací nezmění.

- **int GetROP2**( ) – vrátí aktuální kreslicí režim pera, viz *tab. 4.5*,
- **int SetROP2**(int nDrawMode) – nastaví aktuální kreslicí režim pera dle parametru **nDrawMode** (viz *tab. 4.5*) a vrátí předchozí kreslicí režim,

*Tab. 4.5 Kreslicí režimy (**nDrawMode**)*

Hodnota	Význam
R2_BLACK	vždy černá barva,
R2_WHITE	vždy bílá barva,
R2_NOP	cíl se nezmění (cíl = cíl),
R2_NOT	cíl se zinveruje (cíl = NOT(cíl)),
R2_COPYPEN	cíl odpovídá barvě pera (cíl = pero),
R2_NOTCOPYPEN	cíl je inverzní k barvě pera (cíl = NOT(pero)),
R2_MERGEPENNOT	cíl = NOT(cíl) OR pero,
R2_MASKPENNOT	cíl = NOT(cíl) AND pero,
R2_MERGENOTPEN	cíl = NOT(pero) OR cíl,
R2_MASKNOTPEN	cíl = NOT(pero) AND cíl,
R2_MERGEPEN	cíl = pero OR cíl,
R2_NOTMERGEPEN	cíl = NOT(pero OR cíl),
R2_MASKPEN	cíl = pero AND cíl,
R2_NOTMASKPEN	cíl = NOT(pero AND cíl),
R2_XORPEN	cíl = pero XOR cíl,
R2_NOTXORPEN	cíl = NOT(pero XOR cíl).

- **COLORREF GetTextColor**( ) – zjistí aktuální barvu fontu,
- **COLORREF SetTextColor**(COLORREF crColor) – nastaví aktuální barvu fontu dle parametru **crColor** a vrátí předchozí barvu.

4.4 ZÁKLADNÍ KRESLICÍ METODY

V tomto odstavci jsou uvedeny základní kreslicí metody. Je pravdou, že třída **CPaintDC** disponuje dalšími metodami, ty ale v tomto úvodním přehledu vynechám.

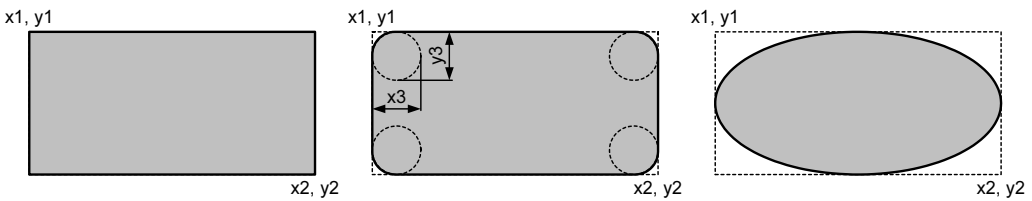
4.4.1 Přesun pera, práce s pixely

- **CPoint GetCurrentPosition**( ) – vrátí aktuální pozici pera jako instanci třídy **CPoint**,
- **CPoint MoveTo**(int x, int y) – přesune pero do bodu **x**, **y**,
- **CPoint MoveTo**(POINT point) – přesune pero do bodu určeného **point**,
- **BOOL LineTo**(int x, int y) – nakreslí čáru z výchozí pozice do bodu **x**, **y**,
- **BOOL LineTo**(POINT point) – nakreslí čáru z výchozí pozice do bodu **point**,
- **COLORREF GetPixel**(int x, int y) – vrátí barvu bodu **x**, **y**,

- **COLORREF GetPixel**(POINT point) – vrátí barvu bodu **point**,
- **COLORREF SetPixel**(int x, int y, COLORREF crColor) – nastaví barvu bodu **x**, **y** na hodnotu **crColor**,
- **COLORREF SetPixel**(POINT point, COLORREF crColor) – nastaví barvu bodu **point** na hodnotu **crColor**.

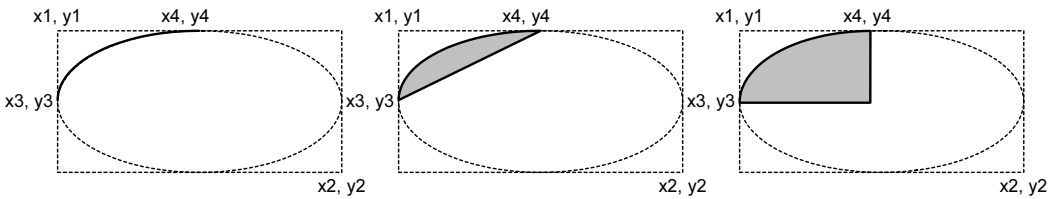
4.4.2 Kreslení obdélníků, elips, oblouků apod.

- **BOOL Rectangle**(int x1, int y1, int x2, int y2) – nakreslí a vyplní obdélník dle souřadnic **x1**, **y1** a **x2**, **y2** (viz *obr. 4.3*),
- **BOOL Rectangle**(LPCRECT lpRect) – nakreslí a vyplní obdélník podle **lpRect**,
- **BOOL RoundRect**(int x1, int y1, int x2, int y2, int x3, int y3) – nakreslí a vyplní obdélník se zakulacenými rohy (viz *obr. 4.3*),
- **BOOL RoundRect**(LPCRECT lpRect, POINT point) – nakreslí a vyplní obdélník se zakulacenými rohy,
- **void FillRect**(LPCRECT lpRect, CBrush* pBrush) – vyplní obdélník určený **lpRect** štětcem **pBrush**,
- **void FrameRect**(LPCRECT lpRect, CBrush* pBrush) – nakreslí obrys obdélníka určeného **lpRect** štětcem **pBrush** (šíře je nyní vždy 1 logická jednotka),
- **void InvertRect**(LPCRECT lpRect) – invertuje obsah obdélníka **lpRect**,
- **void DrawFocusRect**(LPCRECT lpRect) – nakreslí rámeček odpovídající fokusu okolo obdélníka **lpRect**,
- **BOOL Ellipse**(int x1, int y1, int x2, int y2) – nakreslí a vyplní elipsu; body **x1**, **y1** a **x2**, **y2** definují levý horní a pravý dolní roh obdélníka, který je opsán elipse,
- **BOOL Ellipse**(LPCRECT lpRect) – nakreslí a vyplní elipsu, která je vepsána do obdélníka určeného **lpRect**,
- **BOOL Arc**(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) – nakreslí eliptický oblouk z bodu **x3**, **y3** do bodu **x4**, **y4** na elipse vepsané obdélníku s levým horním rohem **x1**, **y1** a pravým dolním rohem **x2**, **y2** (viz *obr. 4.4*),



Obr. 4.3 Obdélník, obdélník se zakulacenými rohy a elipsa

- **BOOL Arc**(LPCRECT lpRect, POINT ptStart, POINT ptEnd) – nakreslí eliptický oblouk z bodu **ptStart** do bodu **ptEnd** na elipse vepsané obdélníku **lpRect**,
- **BOOL Chord**(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) – nakreslí a vyplní eliptickou úseč z bodu **x3**, **y3** do bodu **x4**, **y4** na elipse vepsané obdélníku s levým horním rohem **x1**, **y1** a pravým dolním rohem **x2**, **y2** (viz *obr. 4.4*),
- **BOOL Chord**(LPCRECT lpRect, POINT ptStart, POINT ptEnd) – nakreslí a vyplní eliptickou úseč z bodu **ptStart** do bodu **ptEnd** na elipse vepsané obdélníku **lpRect**,
- **BOOL Pie**(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) – nakreslí a vyplní eliptickou výseč z bodu **x3**, **y3** do bodu **x4**, **y4** na elipse vepsané obdélníku s levým horním rohem **x1**, **y1** a pravým dolním rohem **x2**, **y2** (viz *obr. 4.4*),
- **BOOL Pie**(LPCRECT lpRect, POINT ptStart, POINT ptEnd) – nakreslí a vyplní eliptickou výseč z bodu **ptStart** do bodu **ptEnd** na elipse vepsané obdélníku **lpRect**,



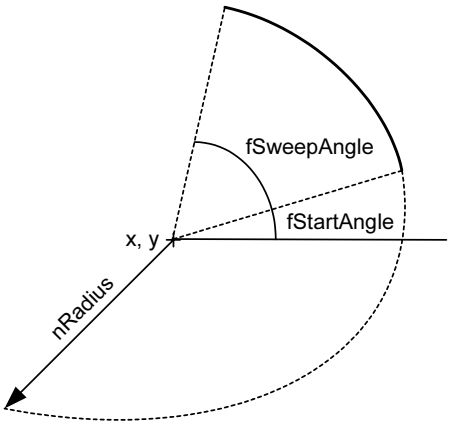
*Obr. 4.4 Metody **Arc**, **Chord** a **Pie***

- **int GetArcDirection**( ) – zjistí orientaci kreslení pro metody **Arc**, **Chord**, **Pie** (viz *tab. 4.6*),
- **int SetArcDirection**(int nArcDirection) – nastaví orientaci kreslení pro metody **Arc**, **Chord**, **Pie** (viz *tab. 4.6*), vrací předchozí orientaci. Funkce je dostupná pouze na platformě Windows NT (Windows 2000/XP),

*Tab. 4.6 Orientace kreslení pro metody **Arc**, **Chord**, **Pie***

Hodnota	Význam
AD_COUNTERCLOCKWISE	obrazec je kreslen proti směru hodinových ručiček (výchozí),
AD_CLOCKWISE	obrazec je kreslen po směru hodinových ručiček.

- **BOOL AngleArc**(int x, int y, int nRadius, float fStartAngle, float fSweepAngle) – nakreslí část kružnice se středem **x**, **y** a poloměrem **nRadius** ze startovacího úhlu **fStartAngle** o velikosti **fSweepAngle** (viz *obr. 4.5*). Počátek oblouku je spojen s aktuálním bodem. Funkce je dostupná pouze na platformě Windows NT (Windows 2000/XP).



*Obr. 4.5 Metoda **AngleArc***

4.4.3 Kreslení mnohoúhelníků a série úseček

- **BOOL Polyline**(LPPOINT lpPoints, int nCount) – nakreslí sérii úseček, které spojují body obsažené v poli **lpPoints**, počet bodů obsažených v poli **lpPoints** je určen parametrem **nCount** (musí to být minimálně 2),
- **BOOL Polygon**(LPPOINT lpPoints, int nCount) – nakreslí mnohoúhelník, vrcholy jsou určené body z pole **lpPoints**, počet bodů (vrcholů) mnohoúhelníka je dán parametrem **nCount**. Pokud není obrazec uzavřený, dojde automaticky ke spojení prvního a posledního bodu.

4.4.4 Kreslení křivek

- **BOOL PolyBezier**(POINT* lpPoints, int nCount) – proloží sérii bodů určenou **lpPoints** Bézierovou křivkou, **nCount** určuje počet bodů. Platí, že počet bodů musí odpovídat vzorci **3×n+1** (tedy například 4, 7, 10, ...).