***GDI***

1 Device Context

1.1 Pribavljanje DC-a

HDC GetDC( HWND *hWnd*)

HDC GetWindowDC( HWND *hWnd*) CDC\* CWnd::GetDC( )

int CWnd::ReleaseDC( CDC\* *pDC* )

1.2 Boja teksta i pozadine teksta

COLORREF CDC::GetTextColor( );

COLORREF CDC::SetTextColor( COLORREF crColor ); COLORREF CDC::GetBkColor( );

COLORREF CDC:: SetBkColor( COLORREF crColor );

1.3 Boja pozadine

int CDC::GetBkMode( );int CDC::SetBkMode(int *iBkMode*);***iBkMode***:

TRANSPARENT

OPAQUE

1.4 Modovi iscrtavanja

int CDC::GetROP2( );int CDC::SetROP2(int *fnDrawMode*);***fnDrawMode***:

R2\_BLACK R2\_COPYPEN R2\_MASKNOTPEN R2\_MASKPEN R2\_MASKPENNOT R2\_MERGENOTPEN R2\_MERGEPEN R2\_MERGEPENNOT R2\_NOP R2\_NOT

R2\_NOTCOPYPEN R2\_NOTMASKPEN R2\_NOTMERGEPENR2\_NOTXORPEN R2\_WHITE R2\_XORPEN

int CDC::GetMapMode( );int CDC:: SetMapMode(int *fnMapMode*); ***fnMapMode***:

MM\_TEXT

MM\_HIENGLISHMM\_HIMETRIC MM\_LOENGLISHMM\_LOMETRICMM\_TWIPS MM\_ISOTROPIC MM\_ANISOTROPIC

BOOL CDC::SetWindowExt(

int *nXExtentW*, // new horizontal window extentint *nYExtentW*, // new vertical window extent );

BOOL CDC:: SetViewportExt(

int *nXExtentV*, // new horizontal viewport extentint *nYExtentV*, // new vertical viewport extent );

BOOL CDC::SetWindowOrg(

int *X*, // new logical x-coordinate of window originint *Y* // new logical y-coordinate of window origin );

BOOL CDC::SetViewportOrg(

int *X*, // new device x-coordinate of viewport originint *Y* // new device y-coordinate of viewport origin );

void CWnd::GetClientRect( LPRECT ***lpRect*** ) const;

1.5 Ispuna poligona

int GetPolyFillMode( HDC *hdc*); int SetPolyFillMode( HDC *hdc*, int *iPolyFillMode*);***iPolyFillMode***:

ALTERNATE WINDING

1.6 Streching modovi

int CDC::GetStretchBltMode( ); int CDC::SetStretchBltMode( int *iStretchMode*); ***iStretchMode***:

BLACKONWHITE COLORONCOLOR HALFTONE

STRETCH\_ANDSCANS - Same as BLACKONWHITE.STRETCH\_DELETESCANS - Same as COLORONCOLOR. STRETCH\_HALFTONE - Same as HALFTONE. STRETCH\_ORSCANS - Same as WHITEONBLACK. WHITEONBLACK

1.7 Snimanje i vraćanje stanja DC-ja

virtual int CDC::SaveDC( );

BOOL CDC::RestoreDC( int *nSavedDC* ); // specifies state to be restored

2 Olovke, četke i crtanje primitiva

2.1 Olovke

CPen::CPen( int *fnPenStyle*, int *nWidth*, COLORREF *crColor*);

***fnPenStyle:***

PS\_SOLID PS\_DASH PS\_DOT PS\_DASHDOTPS\_DASHDOTDOTPS\_NULL PS\_INSIDEFRAME

BOOL CPen::CreatePenIndirect( LPLOGPEN *lpLogPen* );

typedef struct tagLOGPEN { UINT lopnStyle;

POINT lopnWidth; COLORREF lopnColor;} LOGPEN;

CPen::CPen(

int ***nPenStyle***, // PS\_GEOMETRIC, PS\_COSMETIC , ...int ***nWidth***, // pen width

const LOGBRUSH ***\*pLogBrush***, // pointer to structure for brush attributes int ***nStyleCount = 0***, // length of array containing custom style bits const DWORD***\* lpStyle*** // optional array of custom style bits);

typedef struct tagEXTLOGPEN {

UINT ***elpPenStyle***; UINT ***elpWidth***;

UINT ***elpBrushStyle***; COLORREF ***elpColor***;LONG ***elpHatch***;

DWORD ***elpNumEntries***; DWORD ***elpStyleEntry[1]***;} EXTLOGPEN;

***dwPenStyle***:

PS\_GEOMETRIC, PS\_COSMETIC

PS\_ALTERNATE, PS\_SOLID, PS\_DASH, ..., PS\_USERSTYLE, PS\_INSIDEFRAMEPS\_ENDCAP\_ROUND, PS\_ENDCAP\_SQUARE, PS\_ENDCAP\_FLATPS\_JOIN\_BEVEL, PS\_JOIN\_MITER, PS\_JOIN\_ROUND

2.2 Četke

CBrush::CBrush( COLORREF *crColor* );

BOOL CBrush::CreateSolidBrush( COLORREF ***crColor*** );‚

BOOL CBrush::CreateHatchBrush( int ***nIndex***, COLORREF ***crColor*** ); ***fnStyle***:

HS\_BDIAGONALHS\_CROSS HS\_DIAGCROSSHS\_FDIAGONALHS\_HORIZONTALHS\_VERTICAL

BOOL CBrush::CreatePatternBrush( CBitmap\* ***pBitmap*** );

BOOL CBrush::CreateDIBPatternBrush( HGLOBAL ***hPackedDIB***, UINT ***nUsage*** );BOOL CBrush::CreateDIBPatternBrush( const void\* ***lpPackedDIB***, UINT ***nUsage*** );

BOOL CBrush::CreateBrushIndirect( const LOGBRUSH\* ***lpLogBrush*** );

typedef struct tagLOGBRUSH {

UINT ***lbStyle***; // BS\_SOLID, BS\_PATTERN, BS\_HATCHED, ... COLORREF ***lbColor***; // DIB\_PAL\_COLORS, DIB\_RGB\_COLORS LONG ***lbHatch***; // HS\_BDIAGONAL, HS\_CROSS, ...} LOGBRUSH;

CPoint CDC::SetBrushOrg( int ***x***, int ***y*** ); CPoint CDC::SetBrushOrg( POINT ***point*** );CPoint CDC::GetBrushOrg( );

BOOL CGdiObject::UnrealizeObject()

2.3 Stock objekti

CGdiObject\* CDC::SelectStockObject( int ***nIndex*** ); ***nIndex***:

BLACK\_BRUSH, DKGRAY\_BRUSH, GRAY\_BRUSH, HOLLOW\_BRUSH,

LTGRAY\_BRUSH, NULL\_BRUSH, WHITE\_BRUSH, BLACK\_PEN, WHITE\_PEN …

2.4 Tačke

COLORREF GetPixel( HDC *hdc*, int *XPos*, int *nYPos* ); COLORREF SetPixel( HDC *hdc*, int *X*, int *Y*, COLORREF *crColor* ); COLORREF CDC::GetPixel( int *x*, int *y* ); COLORREF CDC::GetPixel( POINT *point* );

COLORREF CDC::SetPixel( int *x*, int *y*, COLORREF *crColor* );COLORREF CDC::SetPixel( POINT *point*, COLORREF *crColor* );

2.5 Linije

BOOL MoveToEx(HDC hdc, int X, int Y, LPPOINT lpPoint);BOOL LineTo(HDC hdc, int nXEnd, int nYEnd); BOOL CDC::LineTo( int *x*, int *y* ); BOOL CDC::LineTo( POINT *point* ); CPoint CDC::MoveTo( int *x*, int *y* ); CPoint CDC::MoveTo( POINT *point* );

BOOL Polyline( HDC *hdc*, CONST POINT ***\*****lppt*, int *cPoints* ); BOOL CDC::Polyline( LPPOINT *lpPoints*, int *nCount* ); BOOL PolylineTo(HDC *hdc*, CONST POINT \**lppt*, DWORD *cCount*); BOOL CDC::PolylineTo( const POINT\* *lpPoints*, int *nCount* ); BOOL PolyPolyline( HDC *hdc*, CONST POINT ***\*****lppt*, CONST DWORD ***\*****lpdwPolyPoints*, DWORD *cCount* );

2.5 Poligon

BOOL Polygon( HDC *hdc*, CONST POINT ***\*****lpPoints*, int *nCount* ); BOOL PolyPolygon( HDC *hdc*, CONST POINT ***\*****lpPoints*, CONST INT ***\*****lpPolyCounts*, int *nCount* );

BOOL Rectangle(HDC *hdc*, int *nLeftRect*, int *nTopRect*, int *nRightRect*, int *nBottomRect*);

BOOL DrawEdge(HDC *hdc*, LPRECT *qrc*, UINT *edge*, UINT *grfFlags*); ***edge***:

BDR\_RAISEDINNER BDR\_SUNKENINNER BDR\_RAISEDOUTER BDR\_SUNKENOUTER***grfFlags***:

BF\_RECT BF\_TOP BF\_LEFT BF\_BOTTOM BF\_RIGHT BF\_TOPLEFT BF\_BOTTOMLEFTBF\_TOPRIGHT BF\_BOTTOMRIGHT BF\_DIAGONAL\_ENDBOTTOMLEFTBF\_DIAGONAL\_ENDBOTTOMRIGHTBF\_DIAGONAL\_ENDTOPLEFT BF\_DIAGONAL\_ENDTOPRIGHT

2.6 Elipse

BOOL Ellipse(HDC *hdc*, int *nLeftRect*, int *nTopRect*, int *nRightRect*, int *nBottomRect*);

2.7 Zaobljeni pravougaonik

BOOL RoundRect( HDC *hdc*, int *nLeftRect*, int *nTopRect*, int *nRightRect*, int nBottomRect, int *nWidth*, int *nHeight* );

2.8 Pite

BOOL Pie(HDC *hdc*, int *nLeftRect*, int *nTopRect*, int *nRightRect*, int *nBottomRect*, int *nXRadial1*, int *nYRadial1*, int *nXRadial2*, int *nYRadial2*);

2.9 Lukovi

BOOL Arc(HDC *hdc*, int *nLeftRect*, int *nTopRect*, int *nRightRect*, int *nBottomRect*, int *nXStartArc*, int *nYStartArc*, int *nXEndArc*, int *nYEndArc*); int SetArcDirection( HDC *hdc*, int *ArcDirection*); ***ArcDirection***:

AD\_COUNTERCLOCKWISE AD\_CLOCKWISE

2.10 Odsečci

BOOL Chord(HDC *hdc*, int *nLeftRect*, int *nTopRect*, int *nRightRect*, int *nBottomRect*, int *nXRadial1*, int *nYRadial1*, int *nXRadial2*, int *nYRadial2*);

2.11 Bezierove krive

BOOL PolyBezier(HDC *hdc*, CONST POINT \**lppt*, DWORD *cPoints*);BOOL PolyBezierTo(HDC *hdc*, CONST POINT \**lppt*, DWORD *cCount*);

3 Regioni, metafajlovi, putanje i transformacije

3.1 Regioni

BOOL CreateRectRgn( int ***x1***, int ***y1***, int ***x2***, int ***y2*** ); BOOL CreateEllipticRgn( int *x1*, int *y1*, int *x2*, int *y2* ); BOOL CreatePolygonRgn( LPPOINT *lpPoints*, int *nCount*, int *nMode* ); //nMode može bitiALTERNATE ili WINDING

int CRgn::CombineRgn( CRgn\* *pRgn1*, CRgn\* *pRgn2*, int *nCombineMode* );

***nCombineMode***: RGN\_AND, RGN\_COPY, RGN\_DIFF, RGN\_OR ili RGN\_XOR

virtual int CDC::SelectClipRgn( CRgn\* *pRgn* );int CDC::SelectClipRgn( CRgn\* *pRgn*, int *nMode* );

BOOL CDC::FloodFill( int *x*, int *y*, COLORREF *crColor* );

3.2 Metafajlovi

CMetaFileDC::CMetaFileDC();

BOOL CMetaFileDC::CreateEnhanced( CDC\* *pDCRef*, LPCTSTR *lpszFileName*, LPCRECT *lpBounds*, LPCTSTR *lpszDescription* ) ;

HENHMETAFILE CMetaFileDC::CloseEnhanced( ) ;

BOOL CDC::DeleteDC();

HENHMETAFILE GetEnhMetaFile( LPCTSTR lpszMetaFile)

HENHMETAFILE CopyEnhMetaFile( HENHMETAFILE *hemfSrc*, LPCTSTR *lpszFile*)

BOOL CDC::PlayMetaFile( HENHMETAFILE hEnhMetaFile, LPCRECT lpBounds)

BOOL DeleteEnhMetaFile( HENHMETAFILE *hemf*)

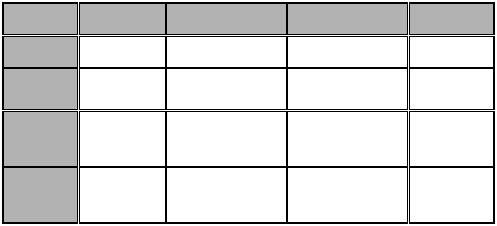
3.3 Putanje

BOOL CDC::BeginPath( ); BOOL CDC::EndPath( );

BOOL CDC::StrokePath( );BOOL CDC::FillPath( );

BOOL CDC:: StrokeAndFillPath( );

BOOL CDC::SelectClipPath( int *nMode* );



***nMode***: RGN\_AND, RGN\_COPY, RGN\_DIFF, RGN\_OR ili RGN\_XOR

3.4 Transformacije

typedef struct \_XFORM { // xfrm

FLOAT eM11; FLOAT eM12; FLOAT eM21; FLOAT eM22; FLOAT eDx; FLOAT eDy;

} XFORM;

operacija eM11 eM12 eM21 eM22

rotacija cos sin -sin cos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skaliranje horizontalno  skaliranje |  | 0 0 vertikalno  skaliranje | | | |
| iskišenje 1 horizontalna  vertikalna  konstanta konstanta  proporcionalnostiproporcionalnosti | | |  | 1 |
| refleksija horizontalna  refleksiona  komponenta |  | 0 0 vertikalna  refleksiona  komponenta | | | | |

BOOL CDC::SetWorldTransform(CONST XFORM *\*lpXform*);

int SetGraphicsMode( HDC hdc, int iMode );

iMode: GM\_COMPATIBLE, GM\_ADVANCED

BOOL ModifyWorldTransform( HDC *hdc,*

CONST XFORM *\*lpXform*, DWORD *iMode*); iMode:

MWT\_IDENTITY – resetuje svetsku transformaciju (učitava se jedinična matrica i ignoriše seprosleđena transf. matrica)

MWT\_LEFTMULTIPLY – množi trenutnu transformacionu matricu sa prosleđenom sa levestrane (prosleđena matrica je levi operand u množenju)

MWT\_RIGHTMULTIPLY – množi trenutnu transformacionu matricu sa prosleđenom sa desne strane (prosleđena matrica je desni operand u množenju)

BOOL CombineTransform( LPXFORM *lpxformResult*, CONST XFORM *\*lpxform1*,

CONST XFORM *\*lpxform2* );

*lpxformResult*– pokazivač na XFORM strukturu koja prihvata kombinovanu transformaciju (rezultujuća matrica)

*lpxform1*– pokazivač na XFORM strukturu prve transformacije (leva matrica)*lpxform2*– pokazivač na XFORM strukturu druge transformacije (desna matrica)

4 Fontovi

4.1 Metrika fonta

BOOL GetTextMetrics( HDC *hdc*, LPTEXTMETRIC *lptm* ); BOOL CDC::GetTextMetrics( LPTEXTMETRIC *lpMetrics* );

typedef struct tagTEXTMETRIC {

LONG ***tmHeight***; LONG ***tmAscent***; LONG ***tmDescent***;

LONG ***tmInternalLeading***; LONG ***tmExternalLeading***; LONG ***tmAveCharWidth***;LONG ***tmMaxCharWidth***; LONG ***tmWeight***; LONG ***tmOverhang***;

LONG ***tmDigitizedAspectX***;LONG ***tmDigitizedAspectY***; BCHAR ***tmFirstChar***; BCHAR ***tmLastChar***; BCHAR ***tmDefaultChar***; BCHAR ***tmBreakChar***; BYTE ***tmItalic***; BYTE ***tmUnderlined***;BYTE ***tmStruckOut***; BYTE ***tmPitchAndFamily***; BYTE ***tmCharSet***;

} TEXTMETRIC;

FW\_DONTCARE 0FW\_THIN 100FW\_EXTRALIGHT 200FW\_ULTRALIGHT 200FW\_LIGHT 300FW\_NORMAL 400FW\_REGULAR 400FW\_MEDIUM 500FW\_SEMIBOLD 600FW\_DEMIBOLD 600FW\_BOLD 700FW\_EXTRABOLD 800FW\_ULTRABOLD 800FW\_HEAVY 900FW\_BLACK 900

***4.2 Tipovi karaktera***

ANSI\_CHARSET DEFAULT\_CHARSETOEM\_CHARSET SYMBOL\_CHARSET

DEFAULT\_PITCHFIXED\_PITCH VARIABLE\_PITCH

FF\_DECORATIVEFF\_DONTCAREFF\_MODERN FF\_ROMAN FF\_SCRIPT FF\_SWISS

HFONT CreateFont (

int ***nHeight***, // logical height of font

int ***nWidth***, // logical average character width int ***nEscapement***, // angle of escapement int ***nOrientation***, // base-line orientation angle int ***fnWeight***, // font weight

DWORD ***fdwItalic***, // italic attribute flag

DWORD ***fdwUnderline***, // underline attribute flag DWORD ***fdwStrikeOut***, // strikeout attribute flag DWORD ***fdwCharSet***, // character set identifier DWORD ***fdwOutputPrecision***, // output precision DWORD ***fdwClipPrecision***, // clipping precisionDWORD ***fdwQuality***, // output quality

DWORD ***fdwPitchAndFamily***, // pitch and family

LPCTSTR ***lpszFace*** // pointer to typeface name string );

HFONT CFont::CreateFont (

int ***nHeight***, // logical height of font

int ***nWidth***, // logical average character width int ***nEscapement***, // angle of escapement int ***nOrientation***, // base-line orientation angle int ***fnWeight***, // font weight

DWORD ***fdwItalic***, // italic attribute flag

DWORD ***fdwUnderline***, // underline attribute flag DWORD ***fdwStrikeOut***, // strikeout attribute flag DWORD ***fdwCharSet***, // character set identifier DWORD ***fdwOutputPrecision***, // output precision DWORD ***fdwClipPrecision***, // clipping precisionDWORD ***fdwQuality***, // output quality

DWORD ***fdwPitchAndFamily***, // pitch and family LPCTSTR ***lpszFace*** // pointer to typeface name string

);

4.3 Veličina ispisanog stringa

CSize CDC::GetTextExtent( const CString& *str* );

4.4 Ispis teksta

BOOL TextOut( HDC *hdc*, int *nXStart*, int *nYStart*, LPCTSTR *lpString*, int *cbString* );BOOL CDC::TextOut( int *x*, int *y*, LPCTSTR *lpszString*, int *nCount* );BOOL CDC::TextOut( int *x*, int *y*, const CString& *str* );

UINT CDC::SetTextAlign( UINT *nFlags* );***nFlags***:

TA\_LEFT TA\_RIGHT TA\_CENTER TA\_TOP TA\_BOTTOM TA\_BASELINE TA\_NOUPDATECPTA\_UPDATECP

int CDC::DrawText( LPCTSTR *lpszString*, int *nCount*, LPRECT *lpRect*, UINT *nFormat* );

int CDC::DrawText( const CString& *str*, LPRECT *lpRect*, UINT *nFormat* ); ***nFormat***:

DT\_BOTTOMDT\_SINGLELINEDT\_CENTERDT\_LEFT DT\_RIGHT DT\_SINGLELINEDT\_TABSTOP DT\_TOP DT\_VCENTER

4.5 Standardni fontovi

CGdiObject\* CDC::SelectStockObject( int *nIndex* ); ***nIndex***:

ANSI\_FIXED\_FONT ANSI\_VAR\_FONT DEVICE\_DEFAULT\_FONTDEFAULT\_GUI\_FONTOEM\_FIXED\_FONT SYSTEM\_FONT

5 Bitmape

5.1 Kreiranje DDB-a

BOOL CBitmap::CreateCompatibleBitmap(CDC\* *pDC*, int *nWidth*, int *nHeight* ); BOOL CBitmap::CreateBitmap( int *nWidth*, int *nHeight*, UINT *nPlanes*, UINT *nBitcount*, const void\* *lpBits* );

5.2 Učitavanje DDB-a

BOOL CBitmap::LoadBitmap( LPCTSTR *pszResName* ); BOOL CBitmap::LoadBitmap( UINT *nIDResource* );

5.3 Pribavljanje bitmape

int CBitmap::GetBitmap( BITMAP\* *pBitMap* );

operator CBitmap::HBITMAP( );

typedef struct tagBITMAP { LONG bmType;

LONG bmWidth; LONG bmHeight; LONG bmWidthBytes; WORD bmPlanes; WORD bmBitsPixel; LPVOID bmBits;} BITMAP;

DWORD CBitmap::SetBitmapBits( DWORD *dwCount*, const void\* *lpBits* );DWORD CBitmap::GetBitmapBits( DWORD *dwCount*, LPVOID *lpBits* );

5.4 Selekcija u DC

CPen\* SelectObject( CPen\* *pPen* ); CBrush\* SelectObject( CBrush\* *pBrush* ); CFont\* SelectObject( CFont\* *pFont* ); CBitmap\* SelectObject( CBitmap\* *pBitmap* ); int SelectObject( CRgn\* *pRgn* );

5.5 Kopiranje bitmape

BOOL CDC::BitBlt( int *x*, int *y*, int *nWidth*, int *nHeight*, CDC\* *pSrcDC*, int *xSrc*, int *ySrc*,

DWORD *dwRop* );***dwRop***:

BLACKNESS DSTINVERT MERGECOPY MERGEPAINT NOTSRCCOPYNOTSRCERASEPATCOPY

PATINVERTPATPAINT SRCAND SRCCOPY SRCERASE

SRCINVERTSRCPAINT WHITENESS

BOOL CDC::PlgBlt( POINT *lpPoint*, CDC\* *pSrcDC*, int *xSrc*, int *ySrc*, int *nWidth*, int *nHeight*, CBitmap& *maskBitmap*, int *xMask*, int *yMask* );

5.6 Rad sa DIB

typedef struct tagBITMAPFILEHEADER {

WORD ***bfType***; DWORD ***bfSize***; WORD ***bfReserved1***; WORD ***bfReserved2***; DWORD ***bfOffBits***;

} BITMAPFILEHEADER;

typedef struct tagBITMAPINFO { BITMAPINFOHEADER ***bmiHeader***; RGBQUAD ***bmiColors[1]***;

} BITMAPINFO;

typedef struct tagRGBQUAD {

BYTE ***rgbBlue***; BYTE ***rgbGreen***; BYTE ***rgbRed***; BYTE ***rgbReserved***;

} RGBQUAD;

typedef struct tagBITMAPINFOHEADER{

DWORD ***biSize***; LONG ***biWidth***; LONG ***biHeight***; WORD ***biPlanes***; WORD ***biBitCount***; DWORD ***biCompression***; DWORD ***biSizeImage***; LONG ***biXPelsPerMeter***; LONG ***biYPelsPerMeter***; DWORD ***biClrUsed***; DWORD ***biClrImportant***;

} BITMAPINFOHEADER;

5.7 Klasa CDib

CDib::CDib();

CDib::CDib(CBitmap& *bitmap*); CDib::CDib(CBitmap\* *bitmap*);DWORD CDib::Width(); DWORD CDib::Height(); WORD CDib::NumColors();

BOOL CDib::Paint(HDC *hDC*, CRect *rcDC*, CRect *rcDIB*); DWORD CDib::Save(CFile& *file*); DWORD CDib::Save(char\* *filename*);DWORD CDib::Read(CFile& *file*); BOOL CDib::Read(char\* *filename*);

5.8 Klasa DImage

DImage(void);

DImage(CBitmap& bmp);virtual ~DImage(void);

bool Load(CString fileName); // Učitava sliku iz datoteke čije se ime navodi bool Save(CString fileName); // Upisuje sliku u datoteku čije se ime navodi void Draw(CDC\* pDC, CRect rcImg, CRect rcDC); // Iscrtava sliku u datom DC-ju

int Width(){return m\_nWidth;} // Širina u pikselimaint Height(){return m\_nHeight;} // Visina u pikselimaint BPP(){return m\_nBPP;} // Broj bajtova po pikselu

bool isValid();

// Direktne izmene

unsigned char\* GetDIBBits(); // Vraća pointer na prvi bajt sa pikselimavoid Update(); // Pozvati nakon direktne izmene bafera.