```
unit Enumeration;

{/-------------------------------------------------------------------------
 Enumeration:
 Enumeriert alle relevanten Informationen der Grafikkarte(n) und
 packt diese in eine Datenstruktur
 Als Vorbild diente der Caps-Viewer aus dem DX-SDK. Wer probleme hat den
 Aufbau der Struktur zu verstehen kann dort unter "DirectX Graphics Adapters"
 nachschauen.
 Eine Vorauswahl irgendwelcher Formate findet nicht statt!

 Copyright 2005 Markus Enkelmann (MexDelphi)
          mexdelphi@gmx.de

 Fragen bitte im Forum http://www.delphidev.de
 http://www.delphidev.de
//-------------------------------------------------------------------------/}

interface

uses
  Direct3D9, Windows, Dialogs, SysUtils ;


//---------------------------------------------------------------------------
// Constanten
//---------------------------------------------------------------------------

// Comstante Device Typen
const d3dDevTypes  : Array[0..2] of D3DDEVTYPE  = (D3DDEVTYPE_HAL,
                                                   D3DDEVTYPE_SW,
                                                   D3DDEVTYPE_REF);

// Constante für die zulässigen Display Modes
const AdapterFormats : Array[0..2] of D3DFORMAT = (D3DFMT_X8R8G8B8,
                                                   D3DFMT_X1R5G5B5,
                                                   D3DFMT_R5G6B5);

// Constante für die zulässigen Backbuffer Modes
const BackBufferFmt : Array[0..5] of D3DFORMAT = (D3DFMT_A2R10G10B10,
                                                  D3DFMT_A8R8G8B8,
                                                  D3DFMT_X8R8G8B8,
                                                  D3DFMT_A1R5G5B5,
                                                  D3DFMT_X1R5G5B5,
                                                  D3DFMT_R5G6B5);

// Depth / Stencil Formate
const DepthStencilFmt : Array[0..5] of D3DFORMAT = (D3DFMT_D16,      //16-bit z-buffer
bit depth.
                                                    D3DFMT_D32,      //32-bit z-buffer bit depth.
```

```
                                    D3DFMT_D24X8,    //32-bit z-buffer bit depth using 24 bits
for the depth channel.
                                    D3DFMT_D24S8,    //32-bit z-buffer bit depth using 24 bits
for the depth channel and 8 bits for the stencil channel.
                                    D3DFMT_D24X4S4,  //32-bit z-buffer bit depth using 24
bits for the depth channel and 4 bits for the stencil channel.
                                    D3DFMT_D15S1);   //16-bit z-buffer bit depth where 15 bits
are reserved for the depth channel and 1 bit is reserved for the stencil channel.

// MultiSample Typen
const MultiSampleTypes : Array[0..16] of D3DMULTISAMPLE_TYPE
=(D3DMULTISAMPLE_16_SAMPLES,
                                        D3DMULTISAMPLE_15_SAMPLES,
                                        D3DMULTISAMPLE_14_SAMPLES,
                                        D3DMULTISAMPLE_13_SAMPLES,
                                        D3DMULTISAMPLE_12_SAMPLES,
                                        D3DMULTISAMPLE_11_SAMPLES,
                                        D3DMULTISAMPLE_10_SAMPLES,
                                        D3DMULTISAMPLE_9_SAMPLES,
                                        D3DMULTISAMPLE_8_SAMPLES,
                                        D3DMULTISAMPLE_7_SAMPLES,
                                        D3DMULTISAMPLE_6_SAMPLES,
                                        D3DMULTISAMPLE_5_SAMPLES,
                                        D3DMULTISAMPLE_4_SAMPLES,
                                        D3DMULTISAMPLE_3_SAMPLES,
                                        D3DMULTISAMPLE_2_SAMPLES,
                                        D3DMULTISAMPLE_NONMASKABLE,
                                        D3DMULTISAMPLE_NONE );

//--------------------------------------------------------------------------------
// Datenstrukturen
//--------------------------------------------------------------------------------

// ~~~Enumeration~~~

// MSInfo : 7. Ebene Multisamle Typ und Multismple Quality
type
  PMSInfo = ^TMSInfo;
  TMSInfo = Record
    MSTyp           : D3DMULTISAMPLE_TYPE;       // multisample type
    MSQuality       : LongWord;                  // Quality Level
  end;

// StencilInfo : 6. Eben, StencilFormat
type
  PStencilInfo = ^TStencilInfo;
  TStencilInfo = Record
    fmt_Stencil     : D3DFORMAT;                 // zu fmt_Adapter und fmt_BackBuffer
passendes Stencil Format
    Anz_MSInfos     : Integer;
```

```
    MSInfo          : Array of TMSInfo          // Multisamples die zum Stencil und Backbuffer
passen
  end;

// BBInfo : 5.Ebene, Ein BackBufferformat das zum AdapterFmt. in ComboInfo passt
type
  PBBInfo = ^TBBInfo;
  TBBInfo = Record
    fmt_BackBuffer    : D3DFORMAT;              // Backbufferformat
    Anz_fmt_Stencil   : Integer;                // wieviele fmt_Stencil sind da
    StencilInfo       : Array of TStencilInfo;  // zu fmt_Adapter und fmt_BackBuffer passende
Stencil Formate
  end;

// ComboInfo : 4. Ebene, Alle passenden Kombinationen aus Front/Back-buffer
type
  PComboInfo = ^TComboInfo;
  TComboInfo = Record
    Windowed          : BOOLEAN;                // Fenster oder Fullscreen
    fmt_Adapter       : D3DFORMAT;              // Adapterformat
    AnzFmt_BackBuffer : Integer;                // Anzahl der Backbuffer für dieses fmt_Adapter
    BBInfo            : Array of TBBInfo;       // Backbuffer für dieses fmt_Adapter
  end;


// DeviceInfo : 3. Ebene, Alle infos zu den devices Hal/SW/REF
type
  PDeviceInfo = ^TDeviceInfo;
  TDeviceInfo = Record
    DevTyp          : D3DDEVTYPE;
    VertProc        : LONGWORD;
    Caps_Device     : D3DCaps9;
    AnzFmt_Adapter  : Integer;
    ComboInfo       : Array of TComboInfo;
  end;

// AdapterInfo : 2. Ebene, Alle infos zum Adapter(Grafikkarte)
type
  PAdapterInfo = ^TAdapterInfo;
  TAdapterInfo = Record
    AnzDispModes    : Integer;
    DispModeSet     : D3DDisplayMode;
    AdIdentif       : D3DADAPTER_IDENTIFIER9;
    DispModes       : Array of D3DDisplayMode;    // Alle Dispmodes die der Adapter
unterstützt
    DeviceInfo      : Array[0..2] of TDeviceInfo; // HAL .. SW .. REF
  end;

// EnumInfo : oberste Ebene
type
  PEnumInfo = ^TEnumInfo;
```

```
  TEnumInfo = Record
    AnzAdapter : Integer;
    Adapter    : Array of TAdapterInfo;
  end;
// ___Enumeration___



// TMaster ist ein Record das alles enthält, was in sämmtlichen Klassen der Engine
// benötigt wird und wird jeder Klasse im Constructor als Pointer übergeben.
// so spart man sich die tausendundein Einzelparameter :)
// hier die etwas verkürzte Version für die Enumeration

// ~~~Main~~~
type
  PMaster = ^TMaster;
  TMaster = Record
    ID3D9          : IDirect3D9;           // Interface um D3D9 objekte zu erstellen
    ID3DDEVICE     : IDirect3DDevice9;     // Interface Device
    PresParams     : TD3DPRESENTPARAMETERS; // PresentParameters zur Device
erstellung
    EnumInfo       : TEnumInfo;            // Datenstruct. alle Enumerierten werte
    IsEnumerated   : BOOLEAN;              // wurde enumeriert
  end;
// ___Main___




type
  TDB_Enumeration = class
  private
    P_Master : PMaster;
    EnumInfo : TEnumInfo;

    function EnumAdapter : HRESULT;
    function EnumCombos(Adapter : Integer)  : HRESULT;

  public
    constructor create(Master : PMaster);
    destructor  destroy; override;

    function Enumerate : HRESULT; override;
  end;

implementation

// ------------------------------------------------------------------------
// Constructor:
// ------------------------------------------------------------------------
```

```
constructor TTDB_Enumeration.create(Master : PMaster);
begin
inherited create;
P_Master := Master;
end;

// ------------------------------------------------------------------------
// Destructor:
// ------------------------------------------------------------------------
destructor TTDB_Enumeration.destroy;
begin
inherited destroy;
end;

// ------------------------------------------------------------------------
// Enumerate:
// 1.Pointer auf das IDirect3D9 interface erzeugen
// 2.Aufruf alle Adapter zu enumerieren
// 3.Aufruf um alle Combos für die Adapter zu enumerieren
// 4.Enumerationsergebnisse in Master speichern
// 5.IDirect3D9 interface wieder freigeben.
// ------------------------------------------------------------------------
function TTDB_Enumeration.Enumerate : HRESULT;
var
   i : Integer;
begin
P_Master.ID3D9 := Direct3DCreate9(D3D_SDK_VERSION);
if P_Master.ID3D9 = nil then
  begin
  Result := S_FALSE;
  exit;
  end;

if EnumAdapter <> S_OK then
  begin
  result := S_FALSE;
  exit;
  end;

for i := 0 to EnumInfo.AnzAdapter -1 do
  begin
  if EnumCombos(i) <> S_OK then
    begin
    result := S_FALSE;
    exit;
    end;
  end;

P_Master.EnumInfo := EnumInfo;
P_Master.IsEnumerated := TRUE;
P_Master.ID3D9 := nil;
```

```
Result := S_OK;
end;


// ------------------------------------------------------------------------
// EnumAdapter:
// Anzahl der Adapter holen (im Normalfall 1 ... hat einer ne Ahnung wie das mit SLI wird ?)
// Loop über die Adapter und alle Relevanten Werte in das record (EnumInfo) speichern
//   - Den Aktuellen DisplayMode speichern
//   - Den Identifier der Grafikkarte (Adapter) speichern
//
//   Loop über alle erlaubten Formate (const AdapterFormats = 3)
//     - Wieviele Modes werden mit diesem Format unterstützt
//
//     Loop über Alle Modes
//       - Display Mode mit dem Format(AdapterFormats) enumerieren und in d3ddspmode
zwischenspeichern
//       - die zahl der Modes um eins erhöhen
//       - Array vergrößern
//       - Speichern von d3ddspmode
//
//     end; // ** mode Loop
//   end; // ** Format Loop
// end; // ** Adapter Loop
// ------------------------------------------------------------------------
function TTDB_Enumeration.EnumAdapter : HRESULT;
var
    AdapterLoop,
    FormatLoop,
    ModeLoop,
    ModeCount   : Integer;
    d3ddspmode  : D3DDISPLAYMODE;

begin
// Anzahl der Grafikkarten
EnumInfo.AnzAdapter := P_Master.ID3D9.GetAdapterCount;
if EnumInfo.AnzAdapter <= 0 then
  begin
  result := S_FALSE;
  exit;
  end;
// Array für die Graka(Adapter) auf richtige länge bringen
SetLength(EnumInfo.Adapter, EnumInfo.AnzAdapter);

//Loop über alle Adapter(Grafikkarten)
for AdapterLoop := 0 to EnumInfo.AnzAdapter -1 do
  begin
  // Den Aktuellen DisplayMode speichern
  if P_Master.ID3D9.GetAdapterDisplayMode(AdapterLoop,
EnumInfo.Adapter[AdapterLoop].DispModeSet) <> D3D_OK then
    begin
```

```
    Result := S_FALSE;
    exit;
    end;
  // Den Identifier der Grafikkarte (Adapter) speichern
  // DeviceName .. DriverVersion etc. sind  dort gespeichert
  if P_Master.ID3D9.GetAdapterIdentifier(AdapterLoop, 0,
EnumInfo.Adapter[AdapterLoop].AdIdentif) <> D3D_OK then
    begin
    Result := S_FALSE;
    exit;
    end;

  //Loop über alle erlaubten Formate (const AdapterFormats = 3)
  for FormatLoop := 0 to 2 do
    begin
    // Wieviele Modes werden mit diesem Format unterstützt
    ModeCount := P_Master.ID3D9.GetAdapterModeCount(AdapterLoop,
AdapterFormats[FormatLoop]);
    // Wenn keiner - ab zum nächsten
    if ModeCount <= 0 then continue;


    //Loop über Alle Modes
    for ModeLoop := 0 to ModeCount -1 do
      begin
      //Display Mode mit dem Format(AdapterFormats) enumerieren und in d3ddspmode
zwischenspeichern
      P_Master.ID3D9.EnumAdapterModes(AdapterLoop, AdapterFormats[FormatLoop],
ModeLoop, d3ddspmode);

      // die zahl der Modes um eins erhöhen
      inc(EnumInfo.Adapter[AdapterLoop].AnzDispModes);
      // Array DispModes[] um eins vergrößern
      SetLength(EnumInfo.Adapter[AdapterLoop].DispModes,
EnumInfo.Adapter[AdapterLoop].AnzDispModes);
      // Speichern von d3ddspmode.

EnumInfo.Adapter[AdapterLoop].DispModes[EnumInfo.Adapter[AdapterLoop].AnzDispMo
des -1] := d3ddspmode;

      end; // ** mode Loop
    end; // ** Format Loop
  end; // ** Adapter Loop

Result := S_OK;
end;



// ------------------------------------------------------------------------
// EnumCombos:
```

```
//
// Loop über die Devices(HAL/SW/REF) des Adapters
//   - Wenn Device da, CAPS speichern
//   - Hardware Support vorhanden ?
//   - VertexProcessing festlegen
//
//   Loop über alle zulässigen Adapter-Formate, hinterlegt in der Constanten AdapterFormats
//
//     Loop Fullscreen/Windowed ( 0 = Wondowed, 1 = Fullscreen)
//       - AdptFmtSwitch setzen, ist TRUE wenn eine neue Combo im Array angelegt werden
muss
//       - bWindowed setzen
//
//       Loop über alle Backbuffer Formate
//         - überprüfen ob eine Combo AdapterFormat<->Backbuffer<->Fullscreen/Windowed
da ist, wenn nein: nächster Schleifwndurchlauf
//         - Wenn eine neue Combo, Platz im Array ComboInfo schaffen
//         - Platz im Array BBInfo schaffen und speichern
//
//         Loop über zulässige Depth/Stencil Formate mit const DepthStencilFmt
//           - Kompatibel mit dem DisplayFormat(fmt_Adapter)
//           - Kompatibel mit dem Backbuffer
//
//           Loop über Multisamples mit const MultiSampleTypes
//             - Multisample mit Backbuffer prüfen
//             - Multisample mit DepthStencil prüfen
//             - Quality Level abfragen
//             - Speichern
//             - Zähler eins rauf
//
//          end; //MSLoop
//        end; //DSLoop
//      end; // Loop BacBufferFormate mit const BackBufferFmt
//    end; // Loop Fullscren Windowed
//  end; // Loop AdapterFormate mit const AdapterFormats//
//end; // Loop Hal .. SW .. REF
//
// Speichern der temporären DeviceInfos in enumInfo
// -----------------------------------------------------------------------
function TTDB_Enumeration.EnumCombos(Adapter : Integer) : HRESULT;
var
   DeviceLoop,                          // Loop Hal .. SW .. REF
   AdptFmtLoop,                          // Loop AdapterFormate mit const AdapterFormats
   WndFuLoop,                            // Loop Fullscren Windowed
   BBFmtLoop,                            // Loop BacBufferFormate mit const BackBufferFmt
   DSLoop,                              // Depth/Stencil Loop
   MSLoop       : Integer;               // Multisample Loop
   bWindowed    : BOOLEAN;                    // temporärer Speicher und Schalter
Windowed/Fullscreen
   tmpDevice    : Array[0..2] of TDeviceInfo;    // temporäre Speicher
   MSQuality    : CARDINAL;                  // MultisampleQualityLevel
```

```
    AdptFmtSwitch : BOOLEAN;                    // Schalter für Speichern


begin
FillChar(tmpDevice, sizeOf(tmpDevice), 0);
// Loop Hal .. SW .. REF
for DeviceLoop := 0 to 2 do
  begin
  // ist Device da ? wenn ja HAL..SW..REF-caps sichern
  if P_Master.ID3D9.GetDeviceCaps(Adapter, D3DDevTypes[DeviceLoop],
tmpDevice[DeviceLoop].Caps_Device )<> D3D_OK then continue;
  // DeviceTyp Speichern
  tmpDevice[DeviceLoop].DevTyp := D3DDevTypes[DeviceLoop];

  // überhaupt HardwareSupport Da ?  (Processing Type)
  // Software geht immer
  tmpDevice[DeviceLoop].VertProc :=
D3DCREATE_SOFTWARE_VERTEXPROCESSING;
  if tmpDevice[DeviceLoop].Caps_Device.DevCaps and
D3DDEVCAPS_HWTRANSFORMANDLIGHT <> 0 then
    begin
    // Hardwareunterstützung da
   tmpDevice[DeviceLoop].VertProc :=
D3DCREATE_HARDWARE_VERTEXPROCESSING;
    end;

  // Loop AdapterFormate mit const AdapterFormats
  for AdptFmtLoop := 0 to 2 do
    begin
    // Loop Fullscren Windowed
    for WndFuLoop := 0 to 1 do
      begin
      AdptFmtSwitch := TRUE; // hier weil windowed rein muss
      if WndFuLoop = 0 then bWindowed := TRUE else bWindowed := FALSE;
      // Loop BackBufferFormate mit const BackBufferFmt
      for BBFmtLoop := 0 to 5 do
        begin


        // Haben wir eine passende combo von AdapterFormat <-> Backbuffer <->
Fullscreen/Windowed ?
        if P_Master.ID3D9.CheckDeviceType(Adapter,
                          D3DDevTypes[DeviceLoop],
                          AdapterFormats[AdptFmtLoop],
                          BackBufferFmt[BBFmtLoop],
                          bWindowed ) <> D3D_OK then continue;

        // Speichern .. Speicher bereitstellen wenn sich Windoewed bzw. AdptFmtSwitch ändern
        if AdptFmtSwitch = TRUE then
          begin
          // AnzFmtAdapter erhöhen
```

```
        inc(tmpDevice[DeviceLoop].AnzFmt_Adapter);
        // Array ComboInfo vergrössern
        SetLength(tmpDevice[DeviceLoop].ComboInfo,
length(tmpDevice[DeviceLoop].ComboInfo)+1);

        // Windowed speichern
        tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].Windowed := bWindowed;
        // fmt_Adapter speichern
        tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].fmt_Adapter := AdapterFormats[AdptFmtLoop];
        AdptFmtSwitch := FALSE;
        end; // AdptFmtSwitch

        // Den rest speichern
        // AnzFmtBB erhöhen
        inc(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer);
        //Array BBInfo vergrössern

SetLength(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo,
Length(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo)+1);
        // BackBufferFmt speichern
        tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].fmt_BackBuffer := BackBufferFmt[BBFmtLoop];



      // Loop über zulässige Depth/Stencil Formate mit const DepthStencilFmt
      for DSLoop := 0 to 5 do
        begin
        // 1.Kompatibel mit dem DisplayFormat(fmt_Adapter)
        // 2.Kompatibel mit dem Backbuffer
        if P_Master.ID3D9.CheckDeviceFormat(Adapter,
                          tmpDevice[DeviceLoop].DevTyp,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].fmt_Adapter,
                          D3DUSAGE_DEPTHSTENCIL,
                          D3DRTYPE_SURFACE,
                          DepthStencilFmt[DSLoop]) <> D3D_OK then continue;

        if P_Master.ID3D9.CheckDepthStencilMatch(Adapter,
                             tmpDevice[DeviceLoop].DevTyp,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].fmt_Adapter,
                             BackBufferFmt[BBFmtLoop],
```

```
                        DepthStencilFmt[DSLoop]) <>D3D_OK then continue;

        // Anz_fmt_Stencil erhöhen
        //inc(tmpDevice[].ComboInfo[].BBInfo[].Anz_fmt_Stencil)
        inc(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil);
        // Array vergrössern
        //SetLength(tmpDevice[].ComboInfo[].BBInfo[].StencilInfo,
Length(tmpDevice[].ComboInfo[].BBInfo[].StencilInfo)+1);

SetLength(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].

BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-1].
              AnzFmt_BackBuffer-1].StencilInfo,

length(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-1].

BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-1].
              AnzFmt_BackBuffer-1].StencilInfo)+1);

        //speichern des DS Formats in combo
        // tmpDevice[].ComboInfo[].BBInfo[].StencilInfo[].fmt_Stencil :=
DepthStencilFmt[DSLoop];
        tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-1].

BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].

StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].

BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].
          Anz_fmt_Stencil -1].fmt_Stencil := DepthStencilFmt[DSLoop];


        //Multisample mit const MultiSampleTypes erfragen
        for MSLoop := 0 to 16 do
          begin
          // Prüfen mit dem BackBuffer
          if (P_Master.ID3D9.CheckDeviceMultiSampleType(Adapter,
                                    tmpDevice[DeviceLoop].DevTyp,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].fmt_BackBuffer,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].Windowed,
```

```
                                    MultiSampleTypes[MSLoop],
                                    @MSQuality) <> D3D_OK)   then continue;
        // Prüfen mit dem Depth/Stencil
        if (P_Master.ID3D9.CheckDeviceMultiSampleType(Adapter,
                                    tmpDevice[DeviceLoop].DevTyp,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-1].fmt_Stencil,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].Windowed,
                                    MultiSampleTypes[MSLoop],
                                    @MSQuality) <> D3D_OK)   then continue;

        //  Quality Level abfragen
        P_Master.ID3D9.CheckDeviceMultiSampleType(Adapter,
                                    tmpDevice[DeviceLoop].DevTyp,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].fmt_BackBuffer,

tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].Windowed,
                                    MultiSampleTypes[MSLoop],
                                    @MSQuality);
        // Platz im Array schaffen

SetLength(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-1].MSInfo,
length(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-1].MSInfo)+1);
        // Speichern
        tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
```

```
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-
1].MSInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-1].Anz_MSInfos].MSQuality := MSQuality-1;
          tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-
1].MSInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-1].Anz_MSInfos].MSTyp :=
MultiSampleTypes[MSLoop];
          // Zähler eins rauf
          inc(tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-
1].StencilInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapt
er-
1].BBInfo[tmpDevice[DeviceLoop].ComboInfo[tmpDevice[DeviceLoop].AnzFmt_Adapter-
1].AnzFmt_BackBuffer-1].Anz_fmt_Stencil-1].Anz_MSInfos);
          end; //MSLoop
        end; //DSLoop
      end; // Loop BacBufferFormate mit const BackBufferFmt
    end; // Loop Fullscren Windowed
  end; // Loop AdapterFormate mit const AdapterFormats
 end; // Loop Hal .. SW .. REF

//Speichern
enumInfo.Adapter[Adapter].DeviceInfo[0] := tmpDevice[0];
enumInfo.Adapter[Adapter].DeviceInfo[1] := tmpDevice[1];
enumInfo.Adapter[Adapter].DeviceInfo[2] := tmpDevice[2];


Result := S_OK;
end;

end.
```