10. Graphics Class Diagram

GfxInterface

```
// Atlas
+ INT AddPageToAtlas();
+ INT AddPageToAtlasFromTgaFileInMemory(
    UINT, unsigned char*);
+ INT AddPageToAtlasFromJpegFile(UINT, CHAR *);


// Font
+ INT AddFont(std::string &,
    std::string &, UINT, UINT);
+ INT SetFont(UINT, std::string &,
    std::string &, UINT, UINT);


// Page
+ INT PageSetup(UINT, std::string &,
    vs_system::ImgBuffer *);
+ UCHAR * PageLock(INT);
+ INT PageUnlock(INT);


// Get buffer dimensions
+ UINT GetDeviceStatus();
+ UINT GetBackBufferWidth();
+ UINT GetBackBufferHeight();

// Get buffer dimensions
+ UINT GetDeviceStatus();
+ UINT GetBackBufferWidth();
+ UINT GetBackBufferHeight();

// Get
+ INT GetClientAdjustedViewrect(
    VS_FloatRectangle *);
+ INT GetClientAdjustedViewport(
    VS_FloatRectangle *);
+ INT GetElementRenderFlag(UINT, BOOL *);
+ INT GetElementRenderMode(UINT, UINT *);
```

```
// Get Element
+ INT  GetElementContext(INT, ElmContext **);
+ UINT GetElementGroupId(INT);
+ UINT GetElementId(INT);


// Set Colour
+ INT SetBackground(VS_Colour *);


// Set
+ INT SetAAQ(UINT);
+ INT SetBorder(VS_FloatRectangle *);
+ INT SetCallbackPickPanel(INT(*)(INT, INT), INT);
+ INT SetCallbackPickElement(INT(*)(INT, INT), INT);
+ INT SetClientCursorPosition(POINT *);
+ INT SetConfigureBitmask(BYTE, std::string);
+ INT SetConfigure(INT);
+ INT SetFPS(UINT);
+ INT SetMargin(VS_FloatRectangle *);
+ INT SetOrthoganalOffCenter(
        FLOAT, FLOAT, FLOAT, FLOAT, FLOAT);
+ INT SetPerspectiveFov(FLOAT, FLOAT, FLOAT);
+ INT SetElementFlag(UINT, BOOL);
+ INT SetElementSwitch(UINT, UINT, BOOL);
+ INT SetSysOutput(SysOutput *);
+ INT SetViewport(VS_FloatRectangle *);
+ INT SetViewportMarkers(UINT, UINT, UINT);
+ INT SetViewrect(VS_FloatRectangle *);
+ INT SetViewRotation(CHAR *, FLOAT, FLOAT, FLOAT);
+ INT SetViewTranslation(FLOAT, FLOAT, FLOAT);


// Create display lists
+ INT AddDisplayList(CHAR *);
+ INT AddElement(INT, CHAR *,
    UINT, UINT, INT(*)(ElmContext *));
+ INT AddElement(INT, CHAR *,
    UINT, UINT, UINT, INT(*)(ElmContext *));
+ INT SetElementParameters(INT, UINT, FLOAT *);
```

```
// Navigate display list
+ INT GetChildCount(INT);
+ INT GetFirst(INT);
+ INT GetNext(INT);


// Render
+ INT RenderBegin();
+ INT RenderViewrect(GfxContext *, BOOL);
+ INT RenderViewport(INT, GfxContext *);
+ INT RenderSetupForViewport(GfxContext *);
+ INT RenderEnd();


// RenderToImage
+ INT RenderToImageBegin();
+ INT RenderToImageEnd();
+ INT RenderToImageEnd(vs_system::ImgBuffer *);
+ INT RenderToImageViewrect(GfxContext *, BOOL);
+ INT RenderToImageViewport(INT, GfxContext *);


// Device Framework
+ INT Initialise(HWND, BOOL, UINT, UINT);
+ INT SetupGraphics();
+ INT CleanupGraphics();
+ INT ResetDevice(UINT, UINT);


// Private Implementation
- class PI_Props;
- PI_Props *pi_props;

- class PI_Element;
- PI_Element *pi_display_list;
```

GfxContext

```
// Configure
+ INT Configure(vs_system::VNode *);

// Default
+ VOID DefaultPerspective();
+ VOID DefaultPerspective(UINT);
+ VOID DefaultOrthogonal();
+ VOID DefaultOrthogonal(UINT);
+ VOID DefaultRotate();
+ VOID DefaultTranslate();


// Get projection ID
+ INT GetTypeId();
```

```
// Get Rect's
+ VS_FloatRectangle *GetMargin();
+ VS_FloatRectangle *GetViewrect();
+ VS_FloatRectangle *GetBorder();
+ VS_FloatRectangle *GetViewport();


// Get Projection parameters
+ FLOAT *GetPerspective();
+ FLOAT *GetOrthogonal();
+ FLOAT *GetZNear();
+ FLOAT *GetZFar();
+ CHAR  *GetView()
+ FLOAT *GetRotate();
+ FLOAT *GetTranslate();
```

```
+ INT  GetName(CHAR *);
+ INT  GetDisplay(CHAR *);
+ INT  GetDisplayListHandle(INT *);
+ INT  GetGroupId(UINT *);
+ INT  GetProcessId(UINT *);
+ UINT GetPanelCTI();
+ BOOL GetPanelBackground();


// Set
+ INT  SetDisplayListHandle(INT);


// Get Render To Image
+ INT  GetRenderToImageDimensions(
    UINT *, UINT *);
+ BOOL GetRenderToImageBegin();
```

```
+ BOOL GetRenderToImageEnd();
+ BOOL GetRenderToImageFlag();
+ INT  GetRenderToImageId();
+ BOOL GetRenderToImageStatus();


// Set Render To Image
+ INT  SetRenderToImageDimensions(
    UINT, UINT);
+ INT  SetRenderToImageFlag(BOOL);
+ INT  SetRenderToImageStatus(BOOL);


// Private Implementation
- class PI_GfxContext;
- PI_GfxContext *pi_gfx_context;
```