Kristýna-GIS

Veřejné rozhraní zásuvných modulů pro vektorové formáty

Specifikace rozhraní verze 1.1

Kristýna-GIS

Uvolněno: Březen 2009
Revize: Duben 2021

Obsah

1. Úvod … 2
2. Použité datové typy … 3
3. Jak to pracuje … 5
  3.1. Přidání vektorových dat do zobrazení … 5
  3.2. Smazání vektorových dat ze zobrazení … 5
  3.3. Vykreslování a tisk vektorových dat … 5
  3.4. Zobrazení vlastností vektorového tématu … 5
  3.5. Nastavení vlastností vektorového tématu … 6
  3.6. Konverze vektorových dat … 6
  3.7. Vytvoření a smazání prostorového indexu … 6
  3.8. Uložení vlastností vektorového tématu do souboru projektu … 6
  3.9. Načtení vlastností vektorového tématu ze souboru projektu … 7
4. Veřejné rozhraní knihovny … 8
  4.1. Minimální funkční požadavky … 9
5. Instalace Vaší vlastní knihovny … 11
  5.1. Identifikační číslo formátu … 11
6. Knihovny poskytované s Kristýnou … 12
7. Příklad … 13
  7.1. Jak příklad zkompilovat … 13
  7.2. Zdrojový kód … 13

1. Úvod

Dokument popisuje rozhraní pro dynamicky připojované knihovny, které může být použito pro čtení a vykreslování prostorových vektorových dat. Rozhraní Vám dovoluje vytvořit Vaší vlastní knihovnu a tímto způsobem můžete rozšířit seznam vektorových formátů, které může Kristýna používat. Pokud vytvoříte svojí vlastní knihovnu, doporučujeme vytvořit také instalační program, který bude knihovnu instalovat. V kapitole *Instalace Vaší vlastní knihovny* naleznete několik doporučení.
Toto je popis rozhraní verze 1.1. Rozhraní je podporováno Kristýnou-GIS verze 5.1 a vyšší. Kristýna-GIS prohlížečka toto rozhraní nepodporuje. Tento materiál popisuje informace dostupné v čase jeho publikování a je v podstatě totožný s informacemi, které jsou obsaženy v nápovědě Kristýny. V některých případech můžete najít aktuálnější informace na internetových stránkách Kristýny (www.christine-gis.com).

2. Použité datové typy

Většina použitých datových typů jsou běžné datové typy používané v Microsoft Windows API.

BOOL - Logický typ (nabývá hodnot TRUE nebo FALSE), je uchováván jako celočíselný typ se znaménkem

BYTE* - Ukazatel na pole bytů

COLORREF - 32-bitová hodnota používaná pro specifikaci RGB barvy. Nejméně významný byte obsahuje hodnotu relativní intenzity červené složky, druhý byte obsahuje hodnotu pro zelenou, a třetí byte obsahuje hodnotu pro modrou. Nejvýznamnější byte musí být nulový. Maximální hodnota jednoho byte je 0xFF.

CRS2CRS – Ukazatel na funkci pro konverzi mezi souřadnicovými systémy. Ukazatel na funkci je deklarován takto:

```
typedef BOOL (__cdecl *CRS2CRS)(int, int, double*, double*, double*)
```

Funkce vrací TRUE pokud je konverze úspěšná, jinak vrací hodnotu FALSE. První parametr funkce je počet bodů ke konverzi. Druhý parametr je posun mezi hodnotami souřadnic v poli souřadnic. Třetí, čtvrtý a pátý parametr jsou ukazatelé na pole souřadnic x, y a z.

double - 64-bitové desetinné číslo se znaménkem

double* - Ukazatel na 64-bitové desetinné číslo se znaménkem

DWORD - 32-bitové celé číslo bez znaménka

EXTENT - Struktura, která definuje rozsah souřadnic X, Y, Z a M hodnot

```
typedef struct _EXTENT {
    double xMin;       // minimální hodnota souřadnice X
    double yMin;       // minimální hodnota souřadnice Y
    double xMax;       // maximální hodnota souřadnice X
    double yMax;       // maximální hodnota souřadnice Y
    double zMin;       // minimální hodnota souřadnice Z (ignorováno)
    double zMax;       // maximální hodnota souřadnice Z (ignorováno)
    double mMin;       // minimální hodnota hodnoty M (ignorováno)
    double mMax;       // maximální hodnota hodnoty M (ignorováno)
} EXTENT;
```

EXTENT* - Ukazatel na strukturu rozsahu

FALSE - Logická hodnota, uložena jako celé číslo se znaménkem s hodnotou nula

HDC - Popisovač kontextu zařízení

HWND - Popisovač okna

int - 32-bitové celé číslo se znaménkem

LONG - 32-bitové celé číslo se znaménkem

NULL - Nulová hodnota

RECT - Struktura definující souřadnice levého horního a pravého spodního rohu obdélníka

```
typedef struct _RECT {
    LONG left;     // x souřadnice levého horního rohu obdélníka
    LONG top;      // y souřadnice levého horního rohu obdélníka
    LONG right;    // x souřadnice pravého dolního rohu obdélníka
    LONG bottom;   // y souřadnice pravého dolního rohu obdélníka
} RECT;
```

RECT* - Ukazatel na strukturu popisující obdélník

TCHAR* - Ukazatel na pole znaků (řetězec)

TRUE - Logická hodnota, uložena jako celé číslo se znaménkem s nenulovou hodnotou

UINT - 32-bitové celé číslo bez znaménka

VECTORFORMATSTRUCT - struktura uchovávající nutné informace pro komunikaci mezi jádrem Kristýny a dynamicky linkovanou knihovnou

```
typedef struct _VECTORFORMATSTRUCT {
    // povinná část struktury
    DWORD    dwSize;    // velikost této struktury (včetně nepovinné části)
    COLORREF mapBkColor;    // barva pozadí mapy
    // nepovinná část struktury
    // obsah nepovinné části závisí na implementaci zásuvného modulu
} VECTORFORMATSTRUCT;
```

void - Použito jako návratový typ funkce specifikuje, že funkce nevrací žádnou hodnotu

void* - Ukazatel na nespecifikovaný datový typ nebo strukturu ("univerzální ukazatel")

3. Jak to pracuje

Kristýna volá funkce veřejného rozhraní popsané v kapitole *Veřejné rozhraní knihovny* dle potřeby. Sada volaných funkcí závisí na typu operace. Knihovna je zavedena do paměti před započetím operace a po ukončení operace je uvolněna. Tato kapitola popisuje všechny podporované operace.

3.1 Přidání vektorových dat do zobrazení

Kristýna najde v registrech Windows dostupné vektorové formáty. Kristýna použije hodnoty z klíčů *FormatName*, *FormatExtensions* a *FormatID* aby vytvořila seznam dostupných formátů a naplní jimi rozbalovací nabídku v dialogovém okně pro přidání tématu. Pokud uživatel zvolí vektorový formát, který používá toto rozhraní, Kristýna přečte v registrech Windows hodnotu klíče *LibFileName* aby byla schopná načíst příslušnou knihovnu. Po načtení knihovny Kristýna zavolá funkci Initialize. Funkce Initialize vrací ukazatel na strukturu VECTORFORMATSTRUCT. Tuto strukturu můžete použít pro uchování ukazatelů na Vaše vlastní objekty, paměťové bloky, dočasné soubory, atd. Kristýny alokuje paměť a uchová tuto strukturu. Struktura je použita ostatními funkcemi, takže ji můžete použít pro sdílení potřebných dat mezi funkcemi.

Seznam volaných funkcí během akce:

```
void* Initialize(TCHAR* fileName)
EXTENT* GetExtent(void* pStruct)
```

3.2 Smazání vektorových dat ze zobrazení

Při mazání tématu ze zobrazení Kristýna volá funkci Release. Ve funkci Release můžete zrušit všechny vytvořené objekty, uvolnit paměť, smazat dočasné soubory, atd.

Seznam volaných funkcí během akce:

```
void Release(void* pStruct)
```

3.3 Vykreslování a tisk vektorových dat

Kristýna nahraje knihovnu do paměti a nastaví hodnotu mapBkColor ve struktuře VECTORFORMATSTRUCT. Poté zavolá funkci Draw. Jakmile je vykreslení hotovo, uvolní knihovnu.

Seznam volaných funkcí během akce:

```
void Draw(HDC hDC, EXTENT* drawExtent, RECT* clipRect, int unitSize, void*
pStruct, CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)
```

3.4 Zobrazení vlastností vektorového tématu

Před tím, než je zobrazen dialog Vlastnosti tématu, Kristýna zavolá sérii funkcí, aby získala všechny potřebné informace.

Seznam volaných funkcí během akce:

```
EXTENT* GetExtent(void* pStruct)
```

```
TCHAR* GetFormatName(void* pStruct)
TCHAR* GetFormatVersion(void* pStruct)
DWORD GetInterfaceVersion()
BOOL HasSpatialIndex(void* pStruct)
BOOL SupportsSpatialIndex()
```

3.5 Nastavení vlastností vektorového tématu

Jakmile je zobrazeno dialogové okno Vlastnosti tématu, uživatel může stisknou tlačítko Změna nastavení v záložce Nastavení. Kristýna zavolá sadu níže uvedených funkcí a dovolí uživateli změnit nastavení zobrazení vektorového souboru v jeho vlastním dialogu.

Seznam volaných funkcí před zobrazením dialogu:

```
TCHAR* GetFormatName(void* pStruct)
void* ChangeSettings(HWND hWndParent, void* pStruct)
```

Seznam volaných funkcí po potvrzení změny nastavení:

```
BOOL HasSpatialIndex(void* pStruct)
DWORD GetInterfaceVersion()
BOOL SupportsReloading()
BOOL NeedsReloading(void* pStruct)
BOOL Reload(void* pStruct)
```

3.6 Konverze vektorových dat

Kristýna najde v registrech Windows hodnotu klíče *LibFileName* a načte patřičnou knihovnu do paměti. Potom zavolá funkci SupportsConversionVectorFormat2Shp nebo SupportConversionShp2VectorFormat podle směru konverze a uvolní knihovnu. Pokud funkce vrátí TRUE, Kristýna použije hodnot klíčů *FormatName*, *FormatExtensions* a *FormatID* a přidá patřičný formát do dialogu pro konverzi. Když uživatel zvolí vektorový formát, který používá toto rozhraní, Kristýna opět přečte hodnotu klíče *LibFileName* v registrech Windows, načte patřičnou knihovnu do paměti a zavolá funkci ConvertVectorFormat2Shp nebo ConvertShp2VectorFormat. Po ukončení konverze knihovnu uvolní.

Seznam volaných funkcí během akce:

```
BOOL ConvertVectorFormat2Shp(TCHAR* shpFileName, TCHAR* vectorFormatFileName,
HWND hWndMainFrame, void* pStruct, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings)
BOOL ConvertShp2VectorFormat(TCHAR* shpFileName, TCHAR* vectorFormatFileName,
HWND hWndMainFrame, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings, BYTE* selection)
DWORD GetInterfaceVersion()
BOOL SupportsConversionVectorFormat2Shp()
BOOL SupportsConversionShp2VectorFormat()
```

3.7 Vytvoření a smazání prostorového indexu

Kristýna najde v registrech Windows hodnotu klíče *LibFileName* a načte patřičnou knihovnu do paměti. Poté zavolá funkci SupportsSpatialIndex. Pokud funkce vrátí TRUE, Kristýna zavolá funkci BuildSpatialIndex nebo RemoveSpatialIndex. Po ukončení akce je knihovna uvolněna.

Seznam volaných funkcí během akce:

```
void BuildSpatialIndex(void* pStruct)
```

```
void RemoveSpatialIndex(void* pStruct)
BOOL SupportsSpatialIndex()
```

3.8 Uložení vlastností vektorového tématu do souboru projektu

Kristýna najde v registrech Windows hodnotu klíče *LibFileName*, načte patřičnou knihovnu do paměti a zavolá funkci WriteSettingsToStringLine. Funkce WriteSettingsToStringLine použije obsah struktury VECTORFORMATSTRUCT pro vytvoření řetězce s nastavením vektorového tématu. Obsah řetězce je závislý na vašich potřebách. Funkce WriteSettingsToStringLine vrací ukazatel na řetězec. Řetězec musí být ukončen binární nulou. Kristýna uloží řetězec do souboru projektu jako hodnotu klíče *vectorThemeSettings*.

Seznam volaných funkcí během akce:

```
TCHAR* WriteSettingsToStringLine(void* pStruct)
```

3.9 Načtení vlastností vektorového tématu ze souboru projektu

Kristýna najde v registrech Windows hodnotu klíče *LibFileName*, načte patřičnou knihovnu do paměti a zavolá funkci ReadSettingsFromStringLine. Funkce ReadSettingsFromStringLine použije řetězec s uloženými hodnotami pro nastavení struktury VECTORFORMATSTRUCT. Obsah řetězce byl vytvořen funkcí WriteSettingsToStringLine (viz kapitola 3.8). Řetězec je ukončen binární nulou. Funkce ReadSettingsFromStringLine vrací ukazatel na změněnou strukturu VECTORFORMATSTRUCT. Pokud selže vrací hodnotu NULL.

Seznam volaných funkcí během akce:

```
void* ReadSettingsFromStringLine(TCHAR* settingsStr, void* pStruct)
```

4. Veřejné rozhraní knihovny

Tato kapitola popisuje veřejné rozhraní pro dynamické knihovny. Každá funkce v tomto rozhraní má přesně definované chování a je volána v situacích popsaných v sekci *Jak to pracuje*.

`void` **`BuildSpatialIndex`**`(void* pStruct)`
Vytvoří prostorový index. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1.

`void*` **`ChangeSettings`**`(HWND hWndParent, void* pStruct)`
Zobrazí dialogové okno s nastavením vektorového souboru. Parametr hWndParent je popisovač rodičovského okna pro dialog. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1. Funkce vrací ukazatel na změněnou strukturu nebo hodnotu NULL pokud je struktura beze změn nebo uživatel stiskl tlačítko Storno.

`BOOL` **`ConvertVectorFormat2Shp`**`(TCHAR* shpFileName, TCHAR* vectorFormatFileName, HWND hWndMainFrame, void* pStruct, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings)`
Konvertuje vektorová data do formátu ESRI shapefile a uloží jej do souboru. Parametr shpFileName je úplný název souboru shapefile (výstupní soubor). Parametr vectorFormatFileName je úplný název vektorového souboru (vstupní soubor). Parametr hWndMainFrame je popisovač hlavního okna Kristýny. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1. Parametr Crs2Crs je ukazatel na funkci pro konverzi mezi souřadnicovými systémy. Pokud není konverze nutná má parametr Crs2Crs hodnotu NULL. Parametr bUseDefaultSetting udává zda by jste pro konverzi dat měli použít implicitní nastavení nebo zobrazit dialog pro změnu nastavení konverze. Funkce vrací hodnotu TRUE pokud je konverze úspěšná, jinak vrací hodnotu FALSE.

`BOOL` **`ConvertShp2VectorFormat`**`(TCHAR* shpFileName, TCHAR* vectorFormatFileName, HWND hWndMainFrame, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings, BYTE* selection)`
Konvertuje ESRI shapefile do vektorového formátu a uloží jej do souboru. Parametr shpFileName je úplný název souboru shapefile (vstupní soubor). Parametr vectorFormatFileName je úplný název souboru vektorového formátu (výstupní soubor). Parametr hWndMainFrame je popisovač hlavního okna Kristýny. Parametr Crs2Crs je ukazatel na funkci pro konverzi mezi souřadnicovými systémy. Pokud není konverze nutná má parametr Crs2Crs hodnotu NULL. Parametr bUseDefaultSetting udává zda by jste pro konverzi dat měli použít implicitní nastavení nebo zobrazit dialog pro změnu nastavení konverze. Parametr selection je ukazatel na pole bytů udávající výběr (nenulová hodnota znamená, že je příslušný záznam vybrán). Můžete použít toto pole pro konverzi pouze vybraných záznamů. Pokud se budou konvertovat všechny záznamy je hodnota parametru selection NULL. Funkce vrací hodnotu TRUE pokud je konverze úspěšná, jinak vrací hodnotu FALSE.

`void` **`Draw`**`(HDC hDC, EXTENT* drawExtent, RECT* clipRect, int unitSize, void* pStruct, CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)`
Vykreslí vektorová data na kontext zařízení s popisovačem hDC. Parametr drawExtent je rozsah pro vykreslení v mapových jednotkách. Parametr clipRect je ořezávací obdélník v jednotkách obrazovky. Parametr unitSize je velikost jedné jednotky v pixelech. Použijte hodnotu tohoto parametru pro stanovení velikosti a šířky symbolů při vykreslování prvků. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1. Parametr Crs2Crs je ukazatel na funkci pro konverzi mezi souřadnicovými systémy. Pokud není konverze nutná má parametr Crs2Crs hodnotu NULL. Parametry pId1 a pId2 jsou ukazatelé na číselné identifikátory. Pokud jsou hodnoty identifikátorů totožné můžete pokračovat ve vykreslování, ale jakmile je identifikátor pId2 změněn měli by jste okamžitě zastavit vykreslování a uvolnit paměť.

`EXTENT*` **`GetExtent`**`(void* pStruct)`
Vrací ukazatel na původní (neprojektovaný) rozsah vektorových dat. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1.

`TCHAR*` **`GetFormatName`**`(void* pStruct)`
Vrací ukazatel na řetězec se jménem vektorového formátu. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1.

`TCHAR*` **`GetFormatVersion`**`(void* pStruct)`

Vrací ukazatel na řetězec s verzí formátu vektorových dat. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1.

`DWORD` **`GetInterfaceVersion`**`()`
Vrací verzi rozhraní. Horní byte obsahuje číslo verze (1) a dolní byte číslo podverze (1).

`BOOL` **`HasSpatialIndex`**`(void* pStruct)`
Vrací hodnotu TRUE pokud mají vektorová data vystavěný prostorový index. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1.

`void*` **`Initialize`**`(TCHAR* fileName)`
Vrací ukazatel na strukturu VECTORFORMATSTRUCT. Pokud se inicializace nezdaří funkce by měla vrátit hodnotu NULL. Kristýna uloží obsah této struktury a použije ji jako parametr při volání ostatních funkcí toto rozhraní. Parametr fileName je úplné jméno souboru s vektorovými daty.

`BOOL` **`NeedsReloading`**`(void* pStruct)`
Vrací hodnotu TRUE pokud musí být vektorová data znovu načtena. Toto může být užitečné když musíte reagovat na nové nastavení již načtených dat. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitoly 3.1 a 3.5.

`void*` **`ReadSettingsFromStringLine`**`(TCHAR* settingsStr, void* pStruct)`
Načte nastavení z řetězce settingsStr při otevírání projektu. Řetězec je ukončen binární nulou. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1. Funkce vrací ukazatel na změněnou strukturu VECTORFORMATSTRUCT. Pokud funkce selže vrací hodnotu NULL.

`void` **`Release`**`(void* pStruct)`
Zruší všechny vytvořené objekty, uvolní paměť, smaže dočasné soubory, atd. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitoly 3.1 a 3.2.

`BOOL` **`Reload`**`(void* pStruct)`
Opětovně načte vektorová data. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitoly 3.1 a 3.5.

`void` **`RemoveSpatialIndex`**`(void* pStruct)`
Smaže prostorový index. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1.

`BOOL` **`SupportsConversionVectorFormat2Shp`**`()`
Vrací hodnotu TRUE pokud knihovna podporuje konverzi z vektorového formátu do shapefile, jinak vrací hodnotu FALSE.

`BOOL` **`SupportsConversionShp2VectorFormat`**`()`
Vrací hodnotu TRUE pokud knihovna podporuje konverzi ze shapefile do vektorového formátu, jinak vrací hodnotu FALSE.

`BOOL` **`SupportsReloading`**`()`
Vrací hodnotu TRUE pokud knihovna podporuje opětovné načtení vektorových dat, jinak vrací hodnotu FALSE.

`BOOL` **`SupportsSpatialIndex`**`()`
Vrací hodnotu TRUE pokud knihovna podporuje prostorovou indexaci, jinak vrací hodnotu FALSE.

`TCHAR*` **`WriteSettingsToStringLine`**`(void* pStruct)`
Vrací ukazatel na řetězec s nastavením vektorového formátu. Tento řetězec je zapsán do souboru projektu. Parametr pStruct je ukazatel na strukturu, která vznikla během inicializačního procesu, viz kapitola 3.1.

4.1 Minimální funkční požadavky

Minimální požadavek je vykreslení vektorových dat v zobrazení a tisk těchto dat na tiskárně. K vytvoření fungující knihovny, která toto splňuje musíte implementovat následující funkce s popsaným chováním:

`void` **`BuildSpatialIndex`**`(void* pStruct)`
Nedělá nic.

`void*` **`ChangeSettings`**`(HWND hWndParent, void* pStruct)`
Zobrazí dialog s informačním hlášením „Žádná nastavení nejsou k dispozici“ a vrátí hodnotu NULL.

`BOOL` **`ConvertVectorFormat2Shp`**`(TCHAR* shpFileName, TCHAR* vectorFormatFileName, HWND hWndMainFrame, void* pStruct, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings)`
Vrátí hodnotu FALSE.

`BOOL` **`ConvertShp2VectorFormat`**`(TCHAR* shpFileName, TCHAR* vectorFormatFileName, HWND hWndMainFrame, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings, BYTE* selection)`
Vrátí hodnotu FALSE.

`void` **`Draw`**`(HDC hDC, EXTENT* drawExtent, RECT* clipRect, int unitSize, void* pStruct, CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)`
Vykreslí vektorová data na kontext zařízení s popisovačem hDC.

`EXTENT*` **`GetExtent`**`(void* pStruct)`
Vrátí ukazatel na rozsah vektorových dat.

`TCHAR*` **`GetFormatName`**`(void* pStruct)`
Vrátí ukazatel na řetězec se jménem datového formátu.

`TCHAR*` **`GetFormatVersion`**`(void* pStruct)`
Vrátí verzi formátu vektorových dat.

`DWORD` **`GetInterfaceVersion`**`()`
Vrátí verzi rozhraní. Horní byte obsahuje číslo verze (1) a dolní byte číslo podverze (1).

`BOOL` **`HasSpatialIndex`**`(void* pStruct)`
Vrátí hodnotu FALSE.

`void*` **`Initialize`**`(TCHAR* fileName)`
Vytvoří objekt VECTORFORMATSTRUCT a vrátí ukazatel na tento objekt.

`BOOL` **`NeedsReloading`**`(void* pStruct)`
Vrátí hodnotu FALSE.

`void*` **`ReadSettingsFromStringLine`**`(TCHAR* settingsStr, void* pStruct)`
Vrátí ukazatel na objekt VECTORFORMATSTRUCT.

`void` **`Release`**`(void* pStruct)`
Nedělá nic.

`BOOL` **`Reload`**`(void* pStruct)`
Vrátí hodnotu FALSE.

`void` **`RemoveSpatialIndex`**`(void* pStruct)`
Nedělá nic.

`BOOL` **`SupportsConversionVectorFormat2Shp`**`()`
Vrátí hodnotu FALSE.

`BOOL` **`SupportsConversionShp2VectorFormat`**`()`
Vrátí hodnotu FALSE.

`BOOL` **`SupportsReloading`**`()`

Vrátí hodnotu FALSE.

`BOOL` **`SupportsSpatialIndex`**`()`
Vrátí hodnotu FALSE.

`TCHAR*` **`WriteSettingsToStringLine`**`(void* pStruct)`
Vrátí ukazatel na řetězec s nulovou délkou.

5. Instalace Vaší vlastní knihovny

Doporučujeme vytvořit instalační program pro instalaci knihovny. Instalační program musí umístit knihovnu do podadresáře dlls v instalačním adresáři Kristýny. Instalační program musí také zapsat následující informace do registrů Windows:

Klíč:
HKEY_LOCAL_MACHINE\Software\Christine-GIS\5.x\VectorFormats\[ID of format]

Hodnoty:

Název	Hodnota	Typ	Max. délka (byte)	
LibFileName	[library.dll]	REG_SZ	255	povinné
FormatExtensions	[*.xxx;*.xxxx]	REG_SZ	63	povinné
FormatName	Název vektorového formátu	REG_SZ	255	povinné
FormatID	[ID formátu]	REG_DWORD	4	povinné

Příklad pro Drawing Interchange File Format:

Klíč:
HKEY_LOCAL_MACHINE\Software\Christine-GIS\5.x\VectorFormats\50

Hodnoty:

Název	Hodnota
LibFileName	dxf.dll
FormatExtensions	*.dxf
FormatName	Drawing Interchange File Format
FormatID	50

Poznámka: Hodnota *FormatID* musí být stejná jako název klíče.

5.1 Identifikační číslo formátu (ID)

Pokud chcete vytvářet vlastní knihovnu pro čtení vektorových dat, ověřte si na webové stránce Kristýny volná identifikační čísla formátu. Poté vyberte volné číslo, kontaktujte nás a my zveřejníme Vámi zvolené číslo na webu Kristýny abychom zabránili problémům s duplicitou. Identifikační číslo formátu je používáno interně Kristýnou a těmito metodami ve skriptovacím jazyce Kristýny: GetType(Number nType) ze třídy Theme, ExportToBmp(String sFullFileName, Number nType, Rect extent, Number nWidth, Number nHeight, Bool bCreateWorldFile) a AddTheme(String sFullFileName, Number nType) ze třídy View, ExportToVectorFormat(String sFullFileName, Number nFormatID, Bool bUseDefaultSettings) ze třídy FTheme, Export(String sFullFileName, Number nOutputType) ze třídy DSCTheme, ShowBmp(String sFullFileName, Number nDelay, Bool bFrame, Number nType) ze třídy MsgBox.

Doporučené rozsahy identifikačních čísle formátů jsou následující:

0 - 1000 pro rastrová a vektorová souborová data (typicky uložena na souborovém systému), rozsah 0 - 100 je rezervován pro Kristýnu-GIS, 101 - 1000 je k dispozici pro vývojáře třetích stran

1001 - 2000 pro rastrová a vektorová data z webovských služeb (typicky generována mapovými servery a zaslána Kristýně), rozsah 1001 - 1100 je rezervován pro Kristýnu-GIS, 1101 - 2000 je k dispozici pro vývojáře třetích stran

2001 - 3000 pro rastrová a vektorová data uložena v geodatabázi (pro budoucí využití, nyní nepodporováno), rozsah 2001 - 2100 je rezervován pro Kristýnu-GIS, 2101 - 3000 je k dispozici pro vývojáře třetích stran

6. Knihovny poskytované s Kristýnou

S Kristýnou-GIS verze 5.1 jsou poskytovány následující knihovny, které mají implementováno toto rozhraní. Tyto knihovny jsou umístěny v podadresáři dlls v instalačním adresáři Kristýny-GIS.

Název formátu:	**Drawing Interchange File Format**
Jméno knihovny:	dxf.dll
ID formátu:	50
Rozšíření souboru:	*.dxf
Verze rozhraní:	1.1
Podpora konverze do shapefile:	Ano
Podpora konverze do vektorového formátu:	Ano
Podpora opětovného načtení:	Ne
Podpora prostorového indexu:	Ano

Název formátu:	**Keyhole Markup Language**
Jméno knihovny:	kml.dll
ID formátu:	51
Rozšíření souboru:	*.kml, *.kmz
Verze rozhraní:	1.1
Podpora konverze do shapefile:	Ano
Podpora konverze do vektorového formátu:	Ano
Podpora opětovného načtení:	Ne
Podpora prostorového indexu:	Ano

Název formátu:	**GPS Exchange Format**
Jméno knihovny:	gpx.dll
ID formátu:	52
Rozšíření souboru:	*.gpx
Verze rozhraní:	1.1
Podpora konverze do shapefile:	Ano
Podpora konverze do vektorového formátu:	Ano
Podpora opětovného načtení:	Ne
Podpora prostorového indexu:	Ano

Název formátu:	**OGS Geopraphy Markup Language**
Jméno knihovny:	gml.dll
ID formátu:	53
Rozšíření souboru:	*.gml, *.xml
Verze rozhraní:	1.1
Podpora konverze do shapefile:	Ano
Podpora konverze do vektorového formátu:	Ano
Podpora opětovného načtení:	Ano
Podpora prostorového indexu:	Ano

7. Příklad

Nabízíme Vám zdrojový kód prázdné kostry pro dynamickou knihovnu používající toto rozhraní. Kód je v jazyce C++ a je testován ve vývojovém prostředí Microsoft Visual Studio 2012.

7.1 Jak příklad zkompilovat

Tato krátká kapitola popisuje kroky vedoucí nutné ke kompilaci zdrojového kódu z kapitoly 7.2 v prostředí Microsoft Visual Studia 2012. Jednoduše následujte instrukce uvedené níže.

- Spusťte Microsoft Visual Studio 2012.
- Z nabídky File zvolte New a poté vyberte příkaz Project. Z kategorií projetků vyberte Win32 v sekci Visual C++. Z instalovaných šablon zvolte Win32 Project. Zvolte umístění pro nový projekt a poté vepište jméno projektu a stiskněte tlačítko OK. V následujícím průvodci zvolte volbu „DLL" a stiskněte tlačítko Finish.
- Je vytvořen projekt, který obsahuje soubor dllmain.cpp.
- Zkopírujte zdrojový kód z kapitoly 7.2. do souboru dllmain.cpp.
- Z nabídky Project zvolte příkaz projectName Properties (Alt + F7). Vyberte sekci Configuration Properties/General a nastavte parametr „Character Set“ na hodnotu „Not Set“. Nyní vyberte sekci Configuration Properties/C/C++/Preprocesor a přidejte do Preprocessor Definitions hodnotu „_CRT_SECURE_NO_WARNINGS“.
- Z nabídky Build zvolte příkaz Build Solution (F7).

7.2 Zdrojový kód

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

typedef BOOL (__cdecl *CRS2CRS)(int, int, double*, double*, double*);

typedef struct _EXTENT {     // rozsah dat
   double xMin;
   double yMin;
   double xMax;
   double yMax;
   double zMin;
   double zMax;
   double mMin;
   double mMax;
} EXTENT;

// mapové jednotky
enum {
    map_units_unknown = 40244,
    map_units_kilometers = 40245,
    map_units_meters = 40246,
    map_units_decimeters = 40253,
    map_units_centimeters = 40247,
    map_units_millimeters = 40248,
    map_units_inches = 40251,
    map_units_feet = 40250,
    map_units_yards = 40254,
    map_units_statute_miles = 40249,
    map_units_nautical_miles = 40255,
    map_units_fathoms = 40256,
    map_units_chains = 40257,
    map_units_links = 40258,
    map_units_us_surveyors_inches = 40259,
    map_units_us_surveyors_feet = 40260,
    map_units_us_surveyors_yards = 40261,
    map_units_us_surveyors_chains = 40262,
```

```
    map_units_us_surveyors_statute_miles = 40263,
    map_units_indian_feet = 40264,
    map_units_indian_yards = 40265,
    map_units_indian_chains = 40266,
    map_units_decimal_degrees = 40252
};

typedef struct _VECTORFORMATSTRUCT {
    // povinná část struktury
    DWORD dwSize;    // velikost této struktury (včetně nepovinné části)
    COLORREF mapBkColor;    // barva pozadí mapy
    // nepovinná část struktury
    // obsah nepovinné části závisí na implementaci zásuvného modulu
    TCHAR fileName[_MAX_PATH];
} VECTORFORMATSTRUCT;

// proměnné pro každé vlákno
typedef struct _THREADPARAMS {
    TCHAR buffer[sizeof(VECTORFORMATSTRUCT)];
    EXTENT extent;
} THREADPARAMS;

HINSTANCE hInstanceDll;
static DWORD dwTlsIndex; // adresa sdílené paměti


BOOL WINAPI DllMain(HINSTANCE hInstDll, DWORD fdwReason, LPVOID lpReserved)
{
    LPVOID lpvData;
    // Podle důvodu volání proveďte patřičnou akci.
    switch(fdwReason)
    {
        case DLL_PROCESS_ATTACH:
            // Inicializace pro každý nový proces.
            // Vrací FALSE pokud natažení DLL selže.

            // Uložte si popisovač instance knihovny
            hInstanceDll = hInstDll;

            // alokuji TLS index.
            if ((dwTlsIndex = TlsAlloc()) == TLS_OUT_OF_INDEXES)
                return FALSE;
            // žádný break: Inicializuji index pro první vlákno.

        case DLL_THREAD_ATTACH:
            // Inicializace pro vlákno.
            // Inicializuji TLS index pro toto vlákno.
            lpvData = (LPVOID)new THREADPARAMS;
            if (lpvData == NULL) return FALSE;
            if (!TlsSetValue(dwTlsIndex, lpvData)) {
                delete (THREADPARAMS*)lpvData;
                return FALSE;
            }
            break;

        case DLL_THREAD_DETACH:
            // Úklid pro vlákno.
            // Uvolním alokovanou paměť pro toto vlákno.
            lpvData = TlsGetValue(dwTlsIndex);
            if (lpvData != NULL) {
                THREADPARAMS* threadParams = (THREADPARAMS*)lpvData;
                if (threadParams) delete threadParams;
            }
            break;

        case DLL_PROCESS_DETACH:
            // Úklid pro proces.
            lpvData = TlsGetValue(dwTlsIndex);
            if (lpvData != NULL) {
                THREADPARAMS* threadParams = (THREADPARAMS*)lpvData;
                if (threadParams) delete threadParams;
            }
            // Uvolním TLS index.
            TlsFree(dwTlsIndex);
            break;
    }
    return TRUE;
}
```

```
// Funkce vrací ukazatel na parametry pro aktuální vlákno
THREADPARAMS* GetThreadParams()
{
    THREADPARAMS* threadParams = (THREADPARAMS*)TlsGetValue(dwTlsIndex);
    if (!threadParams) {
        threadParams = new THREADPARAMS;
        if (!threadParams) return NULL;
        ZeroMemory(threadParams, sizeof(THREADPARAMS));
        if (!TlsSetValue(dwTlsIndex, (LPVOID)threadParams)) {
            delete threadParams;
            return NULL;
        }
    }
    return threadParams;
}

extern "C" __declspec( dllexport ) void __cdecl BuildSpatialIndex(void* pStruct)
{
    // Vytvořte prostorový index
    // ...
    // zde vložte Váš kód
    // ...
    return;
}

extern "C" __declspec( dllexport ) void* __cdecl ChangeSettings(HWND hWndParent, void* pStruct)
{
    THREADPARAMS* threadParams = GetThreadParams();
    if (!threadParams) return NULL;
    memcpy(threadParams->buffer, pStruct, ((VECTORFORMATSTRUCT*)pStruct)->dwSize);
    // Upravte kopii struktury pStruct a vraťte ukazatel na upravenou strukturu
    // ...
    // zde vložte Váš kód
    // ...
    // return threadParams->buffer;

    ::MessageBox(hWndParent, "Žádná nastavení nejsou k dispozici.", "Christine-GIS",
MB_OK|MB_ICONINFORMATION);
    return NULL;
}

extern "C" __declspec( dllexport ) BOOL __cdecl ConvertVectorFormat2Shp(TCHAR* shpFileName, TCHAR*
vectorFormatFileName, HWND hWndMainFrame, void* pStruct, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings)
{
    // Konvertujte vektorový formát do ESRI shapefile a uložte ho do souboru
    // ...
    // zde vložte Váš kód
    // ...
    return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl ConvertShp2VectorFormat(TCHAR* shpFileName, TCHAR*
vectorFormatFileName, HWND hWndMainFrame, CRS2CRS Crs2Crs, BOOL bUseDefaultSettings, BYTE*
selection)
{
    // Konvertujte ESRI shapefile do vektorového formatu and save it to a file
    // ...
    // zde vložte Váš kód
    // ...
    return FALSE;
}

extern "C" __declspec( dllexport ) void __cdecl Draw(HDC hDC, EXTENT* drawExtent, RECT* clipRect,
int unitSize, void* pStruct, CRS2CRS Crs2Crs, DWORD* pId1, volatile DWORD* pId2)
{
    // Funkce běží dokud *pId1 je rovno *pId2
    if (*pId1 != *pId2) return;

    THREADPARAMS* threadParams = GetThreadParams();
    if (!threadParams) return;

    FILE* f = fopen(((VECTORFORMATSTRUCT*)pStruct)->fileName, "rb");
    // ...
    // zde vložte Váš kód
    // ...
    // Prosím nezapomeňte na hodnotu pId2. Kontrolujte tuto hodnotu
    // tak často jak je to možné a rozumné. Pokud je hodnota změněna
```

```
    // okamžitě uvolněte paměť a opusťte funkci.
    fclose(f);
}

extern "C" __declspec( dllexport ) EXTENT* __cdecl GetExtent(void* pStruct)
{
    THREADPARAMS* threadParams = GetThreadParams();
    if (!threadParams) return NULL;
    memset(&(threadParams->extent), 0, sizeof(EXTENT));

    FILE* f = fopen(((VECTORFORMATSTRUCT*)pStruct)->fileName, "rb");
    // Naplňte tyto členy struktury extent:
    // xMin, yMin, xMax a yMax
    // ...
    // zde vložte Váš kód
    // ...
    fclose(f);

    return &(threadParams->extent);
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetFormatName(void* pStruct)
{
    THREADPARAMS* threadParams = GetThreadParams();
    if (!threadParams) return NULL;
    threadParams->buffer[0] = '\0';

    // Naplňte threadParams->m_buffer názvem vektorového formátu
    // ...
    // zde vložte Váš kód
    // ...

    return threadParams->buffer;
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetFormatVersion(void* pStruct)
{
    THREADPARAMS* threadParams = GetThreadParams();
    if (!threadParams) return NULL;
    threadParams->buffer[0] = '\0';

    // Naplňte threadParams->m_buffer verzí vektorového formátu
    // ...
    // zde vložte Váš kód
    // ...

    return threadParams->buffer;
}

extern "C" __declspec( dllexport ) DWORD __cdecl GetInterfaceVersion()
{
    // vraťte verzi rozhraní - 1.0
    return MAKELONG(0, 1);
}

extern "C" __declspec( dllexport ) BOOL __cdecl HasSpatialIndex(void* pStruct)
{
    // ...
    // zde vložte Váš kód
    // ...
    return FALSE;
}

extern "C" __declspec( dllexport ) BYTE* __cdecl Initialize(TCHAR* fileName)
{
    THREADPARAMS* threadParams = GetThreadParams();
    if (!threadParams) return NULL;
    VECTORFORMATSTRUCT* pStruct = (VECTORFORMATSTRUCT*)threadParams->buffer;
    pStruct->dwSize = sizeof(VECTORFORMATSTRUCT);
    // člen mapBkColor ze struktury VECTORFORMATSTRUCT je inicializován
    // Kristýnou předtím než je volána funkce Draw
    lstrcpy(pStruct->fileName, fileName);
    // ...
    // zde vložte Váš kód
    // ...
    return (BYTE*)pStruct;
}
```

```
extern "C" __declspec( dllexport ) BOOL __cdecl NeedsReloading(void* pStruct)
{
    // ...
    // zde vložte Váš kód, vraťte TRUE pokud je vyžadováno opětovné načtení
    // ...
    return FALSE;
}

extern "C" __declspec( dllexport ) void* __cdecl ReadSettingsFromStringLine(TCHAR* settingsStr,
void* pStruct)
{
    THREADPARAMS* threadParams = GetThreadParams();
    if (!threadParams) return NULL;
    memcpy(threadParams->buffer, pStruct, ((VECTORFORMATSTRUCT*)pStruct)->dwSize);
    VECTORFORMATSTRUCT* pVectorStruct = (VECTORFORMATSTRUCT*)threadParams->buffer;
    // Přečtěte nastavení z řetězce settingsStr. Uložte
    // nastavení do pVectorStruct a vraťte ukazatel
    // na pVectorStruct.
    // ...
    // zde vložte Váš kód
    // ...
    // return (BYTE*)pVectorStruct;

    return pVectorStruct;
}

extern "C" __declspec( dllexport ) void __cdecl Release(void* pStruct)
{
    // Vykonejte nutný úklid - zrušte všechny vytvořené objekty,
    // uvolněte paměť, smažte dočasné soubory, atd.
    // ...
    // zde vložte Váš kód
    // ...
    return;
}

extern "C" __declspec( dllexport ) BOOL __cdecl Reload(void* pStruct)
{
    // ...
    // zde vložte Váš kód, pokud je opětovné načtení úspěšné vraťte TRUE
    // ...
    return FALSE;
}

extern "C" __declspec( dllexport ) void __cdecl RemoveSpatialIndex(void* pStruct)
{
    // ...
    // zde vložte Váš kód
    // ...
    return;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsConversionVectorFormat2Shp()
{
    // Pokud není podporováno vraťte FALSE, jinak vraťte TRUE.
    return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsConversionShp2VectorFormat()
{
    // Pokud není podporováno vraťte FALSE, jinak vraťte TRUE.
    return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportReloading()
{
    // Pokud není podporováno vraťte FALSE, jinak vraťte TRUE.
    return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl SupportsSpatialIndex()
{
    // Pokud není podporováno vraťte FALSE, jinak vraťte TRUE.
    return FALSE;
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl WriteSettingsToStringLine(void* pStruct)
{
```

```
     static TCHAR stringLine[8192];    // 8 kB buffer
     stringLine[0] = '\0';
    // Zapište nastavení ze struktury pStruct do stringLine
    // ...
    // zde vložte Váš kód
    // ...

    return stringLine;
}
```