アップグレードの近道

Migration Paths

4D Developer Conference Tokyo 2012

目次

アップグレードの近道

概要

4D v11/v12/v13 は，過去のバージョンとの互換性をできるだけ維持しながらも，今日の標準規格，将来のオペレーションシステムやビジネスアプリケーションの行方を慎重に考慮し，いくつかの重要な変更点が段階的に実装された計画的なアップグレードです。4D は，アップグレードに伴う心配や苦労をできるだけ吸収することを心がけています。とはいえ，今回のアップグレードが，6.8→2003 などとはまったく別次元のものであることも認識しています。

4D v11 SQL のリリース以降，世界中の 4D デベロッパーがアプリケーションのアップグレードに取り組んできました。20 年以上前に開発されたプログラムが保守されているケースも珍しくありません。比較的，早期にアップグレードを決定したのは，100 クライアント以上のシステム，10GB 以上データベース，Intel Mac 対応，64 ビット Windows 対応などのスケーラビリティ要求があった方々でした。その後，ほかのデベロッパーが続き，昨年までに，さまざまなアプリケーションが v12 にアップグレードされました。とはいえ，v12 への本格的な移行をこれから控えているアプリケーションも少なくありません。

今回のアップグレードには，新しい仕様（Unicode，SQL）の対応，サポートが終了したツール（4D Draw，4D Open）の置き換え，新しいデータベースエンジンとプロセス管理の対応など，数々の課題が伴っているので，一体，どこから手をつければ良いのか，途方に暮れるように感じるかもしれません。そのような場合，ほんとうに変更が必要な箇所を見極め，最小限の変更で解決する方法が分かれば，アップグレードという「道」を完全に避けることはできないとしても，かなりの時間と労力を節約することができます。問題は，その「近道」を誰がどうやって発見するのか，という点です。

今回のセッションでは，アップグレードを困難にしかねない，代表的な問題をいくつか取り上げ，それを越えるための「近道」を紹介してゆきます。いずれも，各国のデベロッパーが実際に採用し，効果的であることが実証されているものです。もちろん，他に方法がないわけではありません。しかし，デベロッパーが抱えている問題は，同じようなものであることが少なくないので，そうであれば，自分で苦労を背負うよりも，最初から皆と同じ「近道」を選択したほうが合理的ではないでしょうか。

アップグレードを手際よく進める上で，欠かせないのは，仕様を正確に把握すること，効果的なツールを活用すること，そして 4D とのコミュニケーションを絶やさないことです。昨年，4D Japan では，これまでの一律的なトレーニングを廃止し，個別にどんな話題でも相談できる「4D Café」を開始しました。このサービスを利用された方々からは「アップグレードに対する不安がなくなった」「開発に対する意欲が高まった」などの積極的な感想が寄せられています。もし，今回のセッションですべての問題が取り上げられなかったのであれば，4D Café に申し込むのはいかがでしょうか。きっと，たくさんのアドバイスが得られるはずです。

4D Cafe

4D Japan オフィスのミーティングルームを「貸し切り」として，パートナーからテクニカルサポートやコンサルティングなど 4D に関するあらゆるご相談を伺います。コーヒーやお茶を飲みつつ，お菓子をつまみながら，カフェでお話しているようにお気楽に，自由に，4D に関することなら何でもどうぞ。その日は弊社のスタッフを独占していただき，ご要望の実現，開発の悩みを解決のために一緒になってお手伝いさせていただきます。4D のバージョンは問いません。アップグレードに関することと無関係でも結構です。

対象：2012 年 4D パートナープログラムにご加入の方（ブロンズ，シルバー，ゴールドのいずれでも可）
費用：無料
開催日：カレンダーをご覧ください。 http:// goo.gl/e3uPt
予約：カレンダーより 4D Cafe の表示がある日をクリックして，参加希望メールをお送りください。
　　先着順でご予約となり，弊社から「予約確定」メールが届きます。
利用時間：午前 10 時より午後 5 時の間であれば1時間でも1日でも可。

Unicode

概要

v12 のデータファイルおよびランゲージの文字モードは UTF-16 です。既存のアプリケーションは，事実上，どれも Shift_JIS を前提に作成されているため，v12 にアップグレードするためには，文字コードに合わせてプログラムを修正しなければなりません。

Unicode モードを有効にするべき？

結論からいえば，Unicode モードを使用しないという選択肢は，完全に問題外であり，すべてのアプリケーションは Unicode モードに移行するべきです。データベースを変換した後，すぐにデータベース設定の「互換性」ページに移動し，Unicode モードを有効にしてください。また「データベース」ページに移動し，データ言語が「日本語」であること，「非文字・非数字のみをキーワード区切り文字とする」「旧バージョン互換の文字列比較を使用する」がいずれも有効であることを確認してください。日本語版を快適に使用するためには，これがベストの組み合わせです。

Unicode モード

アップグレードした直後，Unicode モードは無効ですが，データファイルは，すでに Unicode に変換されていることを忘れないでください。Unicode モードを有効にすることにより，アプリケーションはネイティブモードで動作し，本来のパフォーマンスが発揮できるようになります。具体的には，配列のコピー，検索，並び替え，文字列処理全般の速度が数十倍から数百倍になり，テキスト型の変数やフィールドサイズの制限が取り払われます。また，潤沢なキャッシュサイズが利用できる，64 ビット版のサーバーでも起動できるようになります。Unicode モードは，必ず有効にしてください。

データ言語「日本語」

Unicode モードが有効であれば，どのデータ言語が選択されていたとしても，日本語をデータベースに登録することはできますが，データ言語を「日本語」に設定することにより，最適のテキスト比較アルゴリズムが適用されます。具体的には，第一アルゴリズム（SORT ARRAY，リストボックスのヘッダークリック，ORDER BY，>，<演算子など，並び替えに使用されるもの）に Unicode の TERTIARY STRENGTH を採用し，第二アルゴリズム（FIND IN ARRAY，Position，Replace string，=，#演算子など，等価照合に使用されるもの）に Unicode の SECONDARY STRENGTH が日本語モードで採用されます。特に重要なのは後者です。日本語以外の設定では，等価照合に PRIMARY STRENGTH が採用されるため，その設定で日本語のひらがな/カタカナを処理した場合，濁点・半濁点の有無が無視されるので注意してください。

非文字・非数字のみをキーワード区切り文字とする

テキストフィールドを「レコードに保存する」設定にしている場合，キーワードインデックスを利用することができます。キーワードは，長文を構成する単語ごとに作成され，新しい比較演算子の「%」で検索することができます。ワイルドカードの「@」と併用することもできます。通常，4D は，Unicode のワードブレイクを単語の区切りとみなします。日本語の場合，これはひらがな/カタカナ/漢字の文字種が切り替わるポイントおよび（桁区切りのカンマ等を除く）記号文字を指します。「非文字・非数字のみをキーワード区切り文字とする」は日本語のために追加されたオプションです。このオプションが有効にされているときは，文字種が切り替わるポイントではなく，分かち書きなど，単純な区切りだけでキーワードが決定します。

旧バージョン互換の文字列比較を使用する

Unicode では，データ言語「日本語」の並び替えルールに「standard」と「traditional」が存在します。前者は，長音記号（ー）を直前の文字の母音と等価に解釈します。つまり，「あー」と「ああ」はイコールです。後者は，Shift_JIS 処理系との互換性を重視したルールで，そのような処理は特にしません。「旧バージョン互換の文字列比較を使用する」オプションは，「traditional」を採用するというものです。データベースで日本語を処理する場合，特別な理由がない限り，このオプションを有効にすることが推奨されています。

メソッドを書き換えるときの注意点

前述したように，データベース変換後，まず最初に Unicode モードを有効にすることが推奨されていますが，それは，もう後戻りはできないということではありません。Unicode モードは，いつでも無効にすることができ，実際，非 Unicode モードを上手に活用することは，アップグレードを成功させるための大事なポイントです。具体的には，下記の方法がとても効果的です。

はじめに，明らかに Unicode で動作が変わることが予想されるメソッドを特定し，そのメソッドから手をつけてゆきます。たとえば，SEND PACKET，RECEIVE PACKET コマンドを使用しているメソッドは，それほど複雑ではないはずなので，作業を開始するのに適しています。

コードを Unicode に書き換えるときには，以前のコードを（If～End if）残しておき，Unicode モードが有効ではないときには，そちらが実行されるようにしておきます。ランタイムの Unicode モードは，Get database parameter で調べることができます。SEND PACKET の場合，Unicode モードが有効であれば，USE CHARACTER SET(″Windows-31j″;0) コマンドが実行されるようにコードを追加し，Unicode モードが無効であれば，以前のコードがそのまま実行されるようにしておきます。それぞれのモードでチェックし，同じようにプログラムが動作することを確認してから，次のメソッドに移ります。最初は，ファイル書き出しやファイル読み込みなど，外部との通信を Windows-31j エンコーディングに書き換えることから始めると良いかもしれません。

Substring，Position などの関数は，大抵，何らかのロジックの一環で使用されているものなので，それぞれのコマンドを個別に書き換えるよりも，まとまったブロックとして処理したほうが合理的です。バイト数の概念が不可分のロジックであれば，Unicode と正規表現（Match regex）で処理するよりも，Convert to text/CONVERT FROM TEXT で Windows-31j エンコーディングに変換したほうが簡単です。反対に，バイト数や文字コードの参照が目的のための単なる手段である場合，正規表現を使用すれば，コードが大幅に簡略化できる場合が少なくありません。ある程度の数をこなせば，大体のパターンが分かるようになるはずです。

いずれにしても，Unicode モードと非 Unicode モードを切り替えられるようにストラクチャを構成していれば，データファイルは，どちらのモードでも共通なので，効率的にアップグレードを進めることができます。

Unicode の専門的な知識は必要？

Unicode/ICU/正規表現の深い知識があれば，確かにアップグレードを進める上で有利ですが，むしろ必要なのは，Unicode の視点で現行の 4D ランゲージをしっかり捉えることです。たとえば，バイト＝C_BLOB/文字＝C_TEXT，内部＝UTF-16/外部＝UTF-8 といった基本原則，アスタリスクで動作が変化するコマンド/Unicode モードで動作が変化するコマンド/USE CHARACTER SET が効かないコマンドを是非ともドキュメントで確認するようにしてください。Unicode が実装されたのは，4D v11 SQL のことなので，v12 のリファレンスには，あまり記述がありません。アップグレードの過程で v11 をスキップすることは，特に問題ありませんが，v11 の重要な資料まで読み飛ばすことがないようにしてください。不明な点があれば，すぐテクニカルサポートに相談することが勧められています。

4D Open

概要

4D Open は，データベース同士を連結させるための正式な手段として普及していたため，そのサポートが終了するという決定は，多くのデベロッパにとって寝耳に水のことでした。後継とされているのは，HTTP あるいは SQL 経由でサーバーに接続することですが，4D Open とプロトコルの互換性はなく，既存のプログラムおよびロジックは，大幅な見直しを迫られることになります。また，セレクションの操作など，4D Open の利便性を完全に再現することには，かなり高度なスキルが要求されます。

4D Open は，複数の 4D アプリケーションを連結するために使用されていることが多く，バージョンが別々であることも珍しくありません。たとえ一時的でもそれを停止することは，システムの生命に関わる問題であり，したがって 4D Open から脱却するための「近道」がどうしても必要です。

4D Open コンポーネント

Roger Vogt（スイス）は、4D Open のエミュレーターコンポーネントを作成し，希望するデベロッパーには，無償でライブラリを提供しています。このコンポーネントのおおきな特徴は，4D Open のコマンドセットを忠実に再現していることであり，ソースコードやロジックの見直しが基本的に必要ないという点です。ネットワークレイヤーには，定評ある NTK（Pluggers）を採用しており，同時に 100 接続以上のリクエストを問題なく処理することができ，4D Open に匹敵する平均 0.2-0.5ms のレスポンスタイムを実現しています。コーディングを工夫し，「バッチモード」で一度のネットワークコールで複数のコマンドを送信すれば，4D Open の 2-3 倍の速度が期待できます。作者は，このコンポーネントを武器に，短期間で何件もの 4D Open プロジェクトを v11/v12 にアップグレードすることができました。

コンポーネントは，サーバー側，クライアント側の二部構成です。サーバー側は，TCP/UDP サーバー機能を備えており，これが 4D Open/4D Server の代わりを務めます。クライアント側は，前述したように 4D Open コマンドと同名のメソッド群で構成されており，既存のコードがそのまま流用できるようになっています。

NTK プラグインが 32 ビットであるため，64 ビットサーバーにこのコンポーネントをインストールすることはできませんが，接続されたクライアントにコンポーネントをインストールすることで問題を回避することはでき，実際，これはサーバーの負荷を軽減する意味でも有効であると考えられています。

ライセンス

コンポーネントは，現在，無償で提供されています。NTK プラグインは，有償のサードパーティ製品ですが，最初に購入すれば，サーバーごとに金額は発生しない良心的なライセンスです。なお，NTK を Web サーバーの代わりに使用するために，別途，4D Web サーバーのライセンスも必要ですが，4D Japan では，この目的で NTK を使用される方のために，特別なライセンスを用意したいと考えています。

詳細をご希望の方は，是非，営業部までご相談ください。

4D Draw

概要

アップグレードを困難にする要因に 4D Draw を挙げるデベロッパーは少なくありません。後継に挙げられている SVG は，4D Draw と SVG では，パラダイムが根本的に違うために互換性がなく，既存のプログラムは，どれも全面的に書き換えることが必要です。4D Draw は，ほとんどの場合，アプリケーションの中で欠かせない役割を負っているため，何らかの方法で保守しなければ，システムの価値がおおきく損なわれてしまいます。4D Draw から SVG に移行するための「近道」がどうしても必要です。

v12

4D Draw のアップグレードライセンスは発行されていませんが，v12 では，ライセンスのチェック自体が省略されているので，Windows 版であれば，4D Draw 2004 のプラグインを使用することができます。ただし，メニューバーなど，一部の表示が「文字化け」することが知られています。

SVG コンポーネント

4D Draw の後継に SVG が選ばれたのは，4D Draw によく似ていたからではなく，現在もっとも広範囲にサポートされているフォーマットであり，特定のベンダーに依存しない，将来的にも安心できる標準だからです。言い換えるならば，4D Draw と SVG のフォーマットに共通点はほとんどありません。しかし，図形を描画する，画像を挿入する，といった特定のタスクに分けて考えれば，4D Draw と SVG にはかなりの類似性があることに気づきます。SVG コンポーネントは，SVG を操作するための XML コードをコンパイルしてライブラリにしたものです。コマンド体系とシンタックスは，4D Draw を意識したものになっており，これを使用すれば，特に XML を意識しなくても，SVG ドキュメントを処理することができます。4D Draw から移行するための第一歩は，(XML/SVG を習得するよりも) SVG コンポーネントから始めることです。

エリア制御

4D Draw	SVG コンポーネント （標準コマンド）
DR ADD TO BACKGROUND	-
DR DO COMMAND	-
DR Error	SVG_Read_last_error
DR EVENT FILTER	-
DR EXPERT COMMAND	-
DR EXPERT MODE	-
DR GET AREA BOUNDARY	(OBJECT GET COORDINATES)
DR Get update mode	-
DR GET MOUSE	(GET MOUSE/MOUSEX/MOUSEY)
DR Get zoom	-
DR LAST CLICK	(GET MOUSE/MOUSEX/MOUSEY)
DR Last event	(Form event)
DR MENU STATUS	-
DR ON ERROR	SVG_Set_error_handler
DR ON EVENT	SVGTool_SET_VIEWER_CALLBACK
DR ON MENU	-
DR REDRAW	-
DR RELEASE BACKGROUND	-
DR REMOVE FROM BACKGROUND	-

DR SCROLL DOCUMENT	(SVG SHOW ELEMENT) (OBJECT SET SCROLL POSITION)
DR SET ENTERABLE	(OBJECT SET ENTERABLE)
DR SET UPDATE MODE	-
DR ZOOM	(TRANSFORM PICTURE)

エリアオプション

4D Draw	(標準コマンド)
DR COORDINATES	-
DR DISPLAY OPTIONS	-
DR Get display	-
DR GET DOCUMENT SIZE	(SVG GET ATTRIBUTE)
DR Get draw mode	(OBJECT Get format)
DR GET GLOBAL PREFERENCES	-
DR GET PREFERENCES	-
DR SET DISPLAY	-
DR SET DOCUMENT SIZE	-
DR SET DRAW MODE	(OBJECT SET FORMAT)
DR SET GLOBAL PREFERENCES	-
DR SET PREFERENCES	-

エリア

4D Draw	SVG コンポーネント
DR AREA TO AREA	SVG_Copy
DR AREA TO FIELD	SVG_Export_to_picture SVG_Export_to_XML
DR Area to picture	SVG_Export_to_picture
DR DELETE OFFSCREEN AREA	SVG_CLEAR
DR FIELD TO AREA	SVG_Open_picture
DR NEW DRAWING	SVG_New
DR New offscreen area	SVG_New
DR OPEN DOCUMENT	SVG_Open_file
DR PICTURE TO AREA	SVG_Open_picture
DR SAVE DOCUMENT	SVG_SAVE_AS_PICTURE SVG_SAVE_AS_TEXT

バインド

4D Draw	SVG コンポーネント
DR ACTIVATE BIND	-
DR ADD TO BIND	-
DR DEACTIVATE BIND	-
DR DELETE BIND	-
DR New bind	-
DR REMOVE FROM BIND	-

属性を求める

4D Draw	SVG コンポーネント
DR GET ARC SPECS	SVG_GET_ATTRIBUTES
DR Get attribute lock	-
DR GET BOUNDARY	-
DR Get corner rounding	SVG_GET_ATTRIBUTES
DR GET ENDMARKS	SVG_GET_ATTRIBUTES
DR GET FILL ATTRIBUTES	SVG_GET_ATTRIBUTES
DR Get handle state	-
DR GET HIGHLIGHT	-
DR Get ID	SVG_Get_ID
DR GET LINE ATTRIBUTES	SVG_GET_ATTRIBUTES
DR GET LINE SPECS	SVG_GET_ATTRIBUTES
DR Get name	-
DR Get object type	SVG_Read_element_type
DR GET POLYGON VERTEX	SVG_GET_ATTRIBUTES
DR Get refnum	-
DR Get rotation	SVG_GET_ATTRIBUTES
DR Get text	SVG_Get_text
DR GET TEXT ATTRIBUTES	SVG_GET_ATTRIBUTES
DR Get text width	-

取り込み/書き出し

4D Draw	SVG コンポーネント
DR ARRAY TO ATTRIBUTE	SVG_SET_ATTRIBUTES_BY_ARRAYS
DR Array to polygon	SVG_New_polygon_by_arrays
DR ATTRIBUTE TO ARRAY	SVG_GET_ATTRIBUTES
DR POLYGON TO ARRAY	-

図形作成

4D Draw	SVG コンポーネント （標準コマンド）
DR Draw arc	SVG_New_arc
DR Draw line	SVG_New_line
DR Draw oval	SVG_New_ellipse
DR Draw rectangle	SVG_New_rect
DR Draw text	SVG_New_text SVG_New_tspan SVG_New_textArea SVG_New_vertical_text
DR End polygon	SVG_PATH_CLOSE
DR Objects to bitmap	(SVG EXPORT TO PICTURE)
DR PLACE PICTURE	SVG_New_embedded_image
DR POLYGON CURVE	SVG_PATH_CURVE SVG_PATH_QCURVE
DR POLYGON LINE	SVG_ADD_POINT
DR START POLYGON	SVG_New_polygon SVG_New_path

図形操作

4D Draw	SVG コンポーネント （標準コマンド）
DR ADD TO BITMAP	(COMBINE PICTURES)
DR ALIGN	-
DR Count	SVG_Count_elements
DR DELETE	SVG_DELETE_OBJECT
DR GROUP	-
DR HIDE	SVG_SET_VISIBILITY
DR LOCK	-
DR MOVE	SVG_SET_TRANSFORM_TRANSLATE
DR ROTATE	SVG_SET_TRANSFORM_ROTATE
DR SCALE	SVG_SET_TRANSFORM_SCALE
DR SIZE	SVG_SET_DIMENSIONS
DR UNGROUP	-

図形選択

4D Draw	SVG コンポーネント （標準コマンド）
DR SELECT	(SVG Find element ID by coordinates)
DR SELECT BY ATTRIBUTE	-
DR SELECT BY REGION	(SVG Find element IDs by rect)

プリント

4D Draw	SVG コンポーネント （標準コマンド）
DR PRINT	(Print object)
DR PRINT BACKGROUND	-
DR PRINT FOREGROUND	-
DR PRINT MERGE	-

4D 環境コマンド

4D Draw	SVG コンポーネント （標準コマンド）
DR INSERT EXPRESSION	-
DR INSERT FIELD	-
DR Place field	-
DR SET FORMAT	-

定規

4D Draw	SVG コンポーネント （標準コマンド）
DR ARRAY BASE TO SCALE	-
DR ARRAY SCALE TO BASE	-
DR Base to scale	-
DR GET ORIGIN	-
DR GET RULER	-
DR GET RULER OPTIONS	-

DR Scale to base	-
DR SET ORIGIN	SVG_SET_VIEWBOX
DR SET RULER	-
DR SET RULER OPTIONS	-

属性設定

4D Draw	SVG コンポーネント （標準コマンド）
DR SET ARC SPECS	SVG_SET_ATTRIBUTES
DR SET ATTRIBUTE LOCK	-
DR SET CORNER ROUNDING	SVG_SET_ROUNDING_RECT
DR SET ENDMARKS	SVG_SET_MARKER
DR SET FILL ATTRIBUTES	SVG_SET_FILL_BRUSH SVG_SET_FILL_RULE
DR SET HANDLE STATE	-
DR SET HIGHLIGHT	-
DR SET LINE ATTRIBUTES	SVG_SET_STROKE_BRUSH SVG_SET_STROKE_DASHARRAY SVG_SET_STROKE_LINECAP SVG_SET_STROKE_LINEJOIN SVG_SET_STROKE_MITERLIMIT SVG_SET_STROKE_WIDTH
DR SET LINE SPECS	SVG_SET_ATTRIBUTES
DR SET NAME	-
DR SET POLYGON VERTEX	SVG_SET_ATTRIBUTES
DR SET REFNUM	-
DR SET TEXT	SVG_SET_TEXTAREA_TEXT SVG_APPEND_TEXT_TO_TEXTAREA
DR SET TEXT ATTRIBUTES	SVG_SET_FONT_COLOR SVG_SET_FONT_FAMILY SVG_SET_FONT_SIZE SVG_SET_FONT_STYLE SVG_SET_TEXT_ANCHOR SVG_SET_TEXT_KERNING SVG_SET_TEXT_LETTER_SPACING SVG_SET_TEXT_RENDERING SVG_SET_TEXT_WRITING_MODE

ユーティリティ

4D Draw	SVG コンポーネント （標準コマンド）
DR Calculate area	-
DR Calculate perimeter	-
DR Clipboard to picture	(GET PICTURE FROM PASTEBOARD)
DR Color to index	-
DR COLOR TO RGB	SVG_Color_RGB_from_long SVG_Color_RGB_from_CMYK SVG_Color_RGB_from_HLS
DR Font name	(Font name)
DR Font number	(Font number)
DR Index to color	-

DR PICTURE TO CLIPBOARD	(SET PICTURE TO PASTEBOARD)
DR RGB to color	-

4D Draw→SVG コンバーター

4D Draw のファイル形式は，4D Draw，PICT，MacPaint のいずれかです。4D Draw (.4DW) は，オリジナルの情報がすべて記録されていますが，プロプライアタリなフォーマットなので，4D Draw 以外のアプリケーションでは開くことができません。PICT (.pict) は，Mac OS のネイティブピクチャフォーマットなので，4D Draw 以外のアプリケーションでも開くことができますが，文書をイメージに変換したものなので，オブジェクト属性など，4D Draw 独自の情報は失われています。MacPaint (.PNT) は，ペイントツールなどでサポートされていたフォーマットです。

Apple は，PICT フォーマットを廃止する予定であり，実際，64 ビットアプリケーション (Cocoa) では，PICT 形式を開くことができません。Windows では，QuickTime または Altura Mac2WIn があれば，PICT 形式を開くことができますが，Mac OS 6 のエミュレーターである Altura は，かなり陳腐化しており，QuickTime は，いまではおもに動画を処理するためのライブラリになりました。また，前述したように，オブジェクト属性が保存されていないという問題があります。そのようなわけで，速い時期に既存の 4D Draw (.4DW/PICT) 文書を SVG に変換することが求められています。

hmReports

hmReports は，Heubach Media (ドイツ) が開発した，プロフェッショナルなプラグイン製品です。データベースと連動した高度なレポートを作成/編集/印刷できることに加え，特筆すべきことに，文書フォーマットとして SVG と 4D Draw (4DW/PICT) の両方をサポートしています。つまり，4D Draw 文書をインポートし，SVG 形式でエクスポートすれば，4D Draw→SVG コンバーターとして利用できるということです。この機能を活用すれば，既存の 4D Draw 文書を SVG 形式に変換することができます。

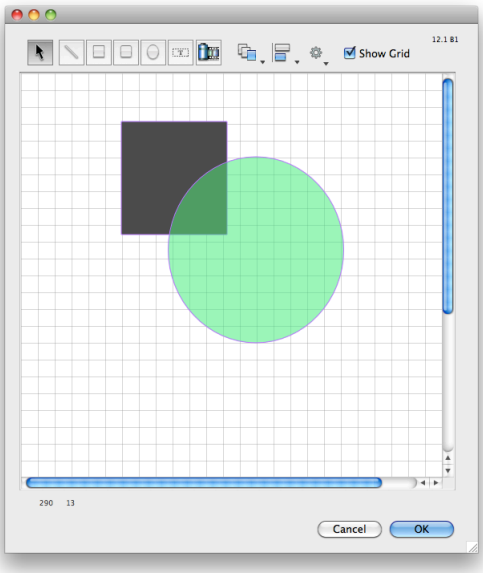
http://www.hmplugins.com/

SVG エリア

4D Draw は，コマンドで図形を描画するだけのプラグインではありません。実際，ほとんどのデベロッパーは，エディター部分を評価し，4D Draw を採用していました。そこで，移行を完了するためには，イタンラクティブな SVG エリアがどうしても必要です。

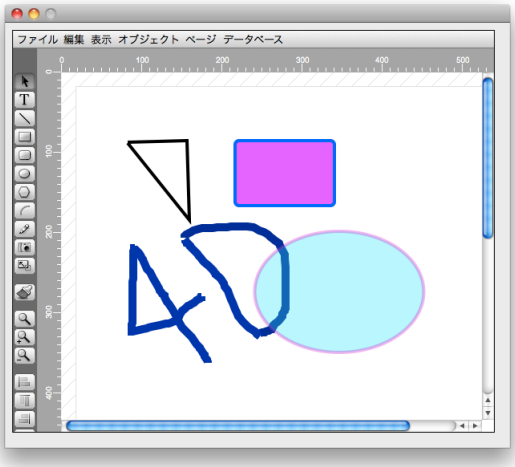
SVG Area (4D)

v12 と一緒に配付されているコンポーネント (ウィジェット) です。インストーラーで「カスタム」を選択することにより，入手することができます。ウィジェットであるため，フォームに組み込むのが容易であり，特別なプログラミングは必要ありません。また，4D コマンドだけで作成されたオープンソースのコンポーネントなので，自由にカスタマイズすることができ，また長期的なサポートが保証されています。



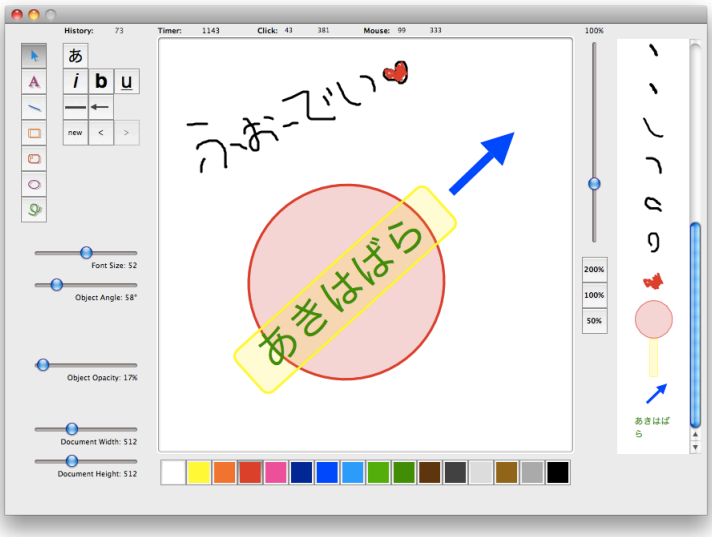
hmReport (Heubach Media)

本来は，レポートを作成するためのツールですが，SVG フォーマットをサポートしており，標準的なツールも揃っているため，ドローエリアとしても通用します。データベースとの連動，ランゲージの埋め込みなど，総合的なスペックは 4D Draw よりもずっと強力です。



SVG Area (sources)

v12 のインストーラーに収録されている SVG Area がそうであるように，標準のコマンドだけでインタラクティブな SVG エリアを作成することができます。sources.4d.com では，プラグインやコンポーネントなど，さまざまな小品が公開されており，最近，ここにオープンソースの SVG エディターも加わりました。



http://sources.4d.com/trac/4d_keisuke/browser/TAM/2012/MIYAKO%20SVG%20Area.4dbase.zip

サポートレベル

4D は，Opera ブラウザの実装を参考に，独自のネイティブ SVG レンダリングエンジンを開発しました。このエンジンは，SVG 1.1 の仕様を基本的にサポートするとともに， SVG tiny 1.2 の仕様を部分的にサポートしています。(要素:textArea, tbreak; 属性:viewport-fill, viewport-fill-opacity, text-align, display-align) また，4D ランゲージとの親和性を考慮し，いくつかのプライベート属性が追加されました。(4D-text, 4D-bringToFront, 4D-isOfClass-*)

それぞれの仕様の詳細は，下記の URL で確認することができます。

http://www.w3.org/TR/SVG11
http://www.w3.org/TR/SVGTiny12/

4D で本格的に SVG を処理する場合，ネイティブ SVG エンジンの実装レベルを厳密に調べたいと思うかもしれません。このエンジンは，これまでに数回(11.0, 11.3, 12.0)改良されてきました。下記に引用したのは，4D v12(2009 年 12 月 7 日)の資料です。

SVG ドキュメント

要素	属性	対応
a		○ g 要素
altGlyph		×
altGlyphDef		×
altGlyphItem		×
animate		×
animateColor		×
animateMotion		×
animateTransform		×
circle	cx cy r	○ ○ ○
clipPath	 clipPathUnits	○ 11.0: rect 12.0: line, polyline, polygon, circle, ellipse, path ○ 11.3
color-profile		×
cursor		×
definition-src		×
defs		○
desc		○
ellipse	cx cy rx ry	○ ○ ○ ○
feBlend	mode in2	○ Windows: normal のみ Mac: normal, multiply, screen, darken, lighten ○
feColorMatrix	type values	○ ○
feComponentTransfer		×
feComposite	operator k1 k2 k3 k4 in2	○ Windows: over のみ Mac: over, in, out, atop × × × × ○
feConvolveMatrix		×
feDiffuseLightning		×
feDisplacementMap		×
feDistantLight		×
feFlood		×
feFuncA		×
feFuncB		×
feFuncG		×
feFuncR		×
feGaussianBlur	stdDeviation	○ x, y ともに最初の偏差値のみ

feImage		×
feMerge		×
feMergeNode		×
feMorphology		×
feOffset	dx dy	○ ○
fePointLight		×
feSpecularLighting		×
feSpotLight		×
feTile		×
feTurbulence		×
filter		○ ローカル URI のみ
font		×
font-face		×
font-face-format		×
font-face-name		×
font-face-src		×
font-face-uri		×
foreignObject		×
g		○
glyph		×
glyphRef		×
hkern		×
image	x y width height xlink:href	○ ○ ○ ○ ○ 11.0: ローカル URI と外部ファイル 11.3: データ URI
line	x1 y1 x2 y2	○ ○ ○ ○
linearGradient	 x1 y1 x2 y2 gradientUnits gradientTransform spreadMethod xlink:href	○ 11.0: 最初と最後の stop のみ 12.0: すべての stop ○ ○ ○ ○ ○ ○ 12.0 ○ 12.0 ○ ローカル URI のみ
marker	 markerUnits refX refY markerWidth markerHeight preserveAspectRatio viewBox orient	○ 11.3 ○ ○ ○ ○ ○ ○ ○ ○

mask		×
metadata		○
missing-glyph		×
mpath		×
path	d	○ 11.3
	pathLength	×
pattern		○ 12.0
	x	○
	y	○
	width	○
	height	○
	viewBox	○
	preserveAspectRatio	○
	xlink:href	○ ローカル URI のみ
	patternUnits	○
	patternContentUnits	○
	patternTransform	○
polygon	points	○
polyline	points	○
radialGradient		○ 11.0: 最初と最後の stop のみ 12.0: すべての stop
	cx	○
	cy	○
	r	○
	fx	○
	fy	○
	gradientUnits	○
	gradientTransform	○ 12.0
	spreadMethod	○ 12.0
	xlink:href	○ ローカル URI のみ
rect	x	○
	y	○
	width	○
	height	○
	rx	○
	ry	○
script		×
set		×
solidColor	solid-color	○
	solid-opacity	○
stop	offset	○
	stop-color	○
	stop-opacity	○
style		○ CSS2 準拠のインライン表記
	type	○ text/css のみ
	media	○ print, screen, all のみ
svg	x	○
	y	○
	width	○
	height	○
	preserveAspectRatio	○
	viewBox	○
	viewport-fill	○
	viewport-fill-opacity	○

switch		×
symbol	viewBox	○
	preserveAspectRatio	○
tbreak		○ 11.3
text	x	○ 最初の値のみ
	y	○ 最初の値のみ
	dx	○ 最初の値のみ
	dy	○ 最初の値のみ
	rotate	×
	textLength	×
	lengthAdjust	×
textArea		○ 11.3
	x	○
	y	○
	width	○
	height	○
textPath		×
title		○
tref		×
tspan	x	○ 最初の値のみ
	y	○ 最初の値のみ
	dx	○ 最初の値のみ
	dy	○ 最初の値のみ
	rotate	×
	textLength	×
use	x	○
	y	○
	width	○
	height	○
	xlink:href	○ ローカル URI のみ
video		×
view		×
vkern		×
すべての要素	color	○ CSS2 準拠
	class	○ 11.3
	clip	×
	clip-path	○ clipPath 要素のみ
	clip-rule	×
	display	○ visibility 属性
	fill	○
	fill-opacity	○
	fill-rule	○ 12.0
	filter	○ ローカル URI のみ
	id	○
	marker	○ ローカル URI のみ
	marker-start	○ ローカル URI のみ
	marker-mid	○ ローカル URI のみ
	marker-end	○ ローカル URI のみ
	opacity	○
	overflow	×
	shape-rendering	○ 12.0: auto または optimizeSpeed (アンチエイリアス無効)

	stroke	○
	stroke-dasharray	○ 12.0
	stroke-dashoffset	○ 12.0
	stroke-opacity	○
	stroke-linecap	○
	stroke-linejoin	○
	stroke-miterlimit	○ 12.0
	style	○ 11.3
	transform	○
	visibility	○
	xml:base	×
	xml:id	○
	xml:lang	○
	xml:space	×
text, textArea, tspan	text-anchor	○
	text-decoration	○ blink 非サポート textArea は overline 非サポート
	font-size-adjust	×
	kerning	○ 12.0: Windows は textArea 非サポート, lr および tb 方向のみ
	font-stretch	×
	font-variant	×
	letter-spacing	○ ○ 12.0: Windows は textArea 非サポート, lr および tb 方向のみ
	word-spacing	×
	glyph-orientation-horizontal	×
	glyph-orientation-vertical	×
	dominant-baseline	×
	direction	×
	unicode-bidi	×
	writing-mode	○ 12.0
	text-rendering	○ 12.0: auto または optimizeSpeed (アンチエイリアス無効)
フィルター要素	x	×
	y	×
	width	×
	height	×
	in	○ SourceGraphic, SourceAlpha のみ
	result	○

4D プライベート属性（SVG SET ATTRIBUTE/SVG GET ATTRIBUTE）

要素	対応
4D-text	○ 12.0: read/write text, tspan, textArea のみ
4D-bringToFront	○ 12.0: write
4D-isOfClass-{IDENT [[S|COMMA] IDENT]*}	○ 12.0: read すべてのクラスが合致すれば true

XML スタイルシート

要素	対応
type	○ 11.3: text/css のみ
href	○ 11.3: 相対パスまたは絶対パス
media	○ 11.3: print, screen, all のみ

CSS2 シンタックス

要素	対応
!important	○ 11.3: 継承値ではない, 明示的な属性をオーバーライドするために
at-rule@media	○ 11.3: print, screen, all のみ
at-rule@important	○ 11.3: print, screen, all のみ
/ comments *\/*	○ 11.3
<!-- comments -->	×
expressions	×
counters	×
functions	×