☆☆☆ アピールポイント ☆☆☆

十数余年に亘る実践に即した熟練評価者の観点から、
評価研究により編み出したノウハウをフルに活用し、
GDI関数を熟知しながらも、
一般ユーザーの視点に即した 複合要素のあるグレイボックステスト**PKSデータ**

上記データを最大限に発揮し
１アプリケーションとしての干渉を極力抑えることで、
GDI関数の高品位なテストをもたらすことを可能にした **Akanesテストツール**

弊社は
上記の **ベストコンビネーション**評価により
国内･国外／全WindowsOS／小型･大型･複合のあらゆる機種を対象とし、
プリンタドライバ（ファーム）
フォントおよびフォントドライバ
の **高品質GDI描画関数テスト**を提供します。

◆ **卓越したバグ検出歴を誇る豊富な テストデータ ラインアップの一例**：
○ 全サポート用紙サイズ１dot精度レベルの印字領域（論理/物理 Objectを含む ﾊｶﾞｷ～超大判）
○ 現環境における使用可能な全 論理/物理フォントの把握（TrueType/Device/System/User等）
○ 上記全網羅フォントにおける全てのテキスト関連関数テスト（追加OptionalFont可）
○ ラスタオペレーション256完全網羅テスト（ROP2・ROP3・ROP4）
○ スタンプ配置等に反映するテキスト回転からワールド座標変換までの正負方向への任意回転描画
○ 全ビットマップ任意拡縮指定描画（DDB/DIB）

その他 Nup/ポスター/鏡映/カラーマッチング/描画優先位/フォント置換え 等々200アイテム以上

◆ **データ作成のポイント１例**：
○ 解決すべきフォントテキストの留意点抽出
・OS/アプリ種により使用可能なフォントマトリクス環境には差異が生じる
・フォントはアプリに依存するためテストしたいフォントで描画された保証はない
・フォントは種類やデザインが多く描画関数の障害検出に適したものを選択しにくい
・アプリにより描画手段はまちまちでどのテキスト関数のどのパラメータ処理に不具合があるか不明
・アプリによっては作成後のテキストデータはビットマップ化されてしまい
　作成時の障害再現が不可となり真のテストデータとはいえなくなる
↓
＜考案考査＞
↓
◇フォントデータ作成
① 各OSのみの環境下でOS毎の固有フォントを把握
② 全フォントの特徴を細部関数によりデザイン/スタイルの特異性まで把握
③ 各種テキスト関連関数の特色をOSパラメータ差異まで把握
④ 各把握した特色に合わせ障害検知しやすいデータをGDI Scriptにより作成
⑤ 尚且つ描画結果に使用されたフォントが指定フォントであるかをOut関数により取得描画
⑥ CreateFontから各種テキスト関連関数に至る複合テストデータ内容が データ作成時と同様に
　テスト時にスクリプトとして毎回実行されるため 真のテストデータ状態を持続し障害再現も容易
↓
＜α版β版での実テスト＞＜各顧客開発とのＱ＆Ａ＞＜対象の修正確認＞＜市場クレーム反映＞
↓
＜更なる研究による改良改善＞　※Bitmapや他の描画Objectに関しても各々の研究作成ステップ

◎ お客様のニーズに合わせた評価および評価データの提供：
- ◆ お手持ちの評価ツールに合わせPKSのデータ内容を強化反映
- ◆ オリジナルデータを駆使した低コスト高品質評価
- ◆ 既知の設計ウィークポイントをカバーするための１ポイント評価
- ◆ 独自の仕様の整合性を確認するための評価データ 等々

◎ ユーザー・ベネフィット例：（対象機処理能力差異あり）

例１：全テスト工数：４０%削減

例２：導入前－１OS x アプリ多数　オペレータ３名～　/５日～

⇒　導入後－１OS x Akanes１　オペレータ１名　/３～５日に短縮

例３：導入前に比べてテスト効率が３５%アップ　等々

◎　多くのお客様が抱えておいでの設計／評価 各フェーズにおける障害検出の課題カバー例

■ GDIを意識していないブラックボックステストデータのみで
　バグ検出させるには多くの時間と評価データを要する

■ アプリのサンプル色が濃いブラックボックステストデータのみでは
　アプリが得意とする描画に留まるためバグが検出されにくい

■ 不具合が検出されても
　アプリがいづれの関数を使用しているか判断しかねる

■ アプリで作成時に検出されたテキストの不具合が
　なぜか再現しないでサービスクレームが懸念される

■ 設計フェーズで関数評価はしているし
　評価フェーズでも実アプリ評価は一巡しているのに潜在バグがある

■ もっと短期間でクオリティは落とさずに
　コストパフォーマンスにすぐれた問題検出方法はないのだろうか

■ とても素晴らしいユニークな仕様を考えたが
　アプリへの整合性や成果を目視確認しながら作り込んでいく方法はないか

■ 印字領域や描画領域に不具合がありそうだが
　アプリ依存するのでもうひとつ問題点が把握できない

■ ビットマップに問題はあるようだが
　どのROP(ラスタオペレーション)が要因なのか的確に検出しにくい

■ デバイスフォントは全てテストしておきたいが
　アプリ依存が否めず確実にテストを網羅したとは言い切れない

お悩みはなんですか？　。。。是非、お手伝いさせてください。

【ご参考】お客様からの声 一例： （各メーカー様 各プロジェクト 開発-部長/課長/担当 及び 評価-課長/担当 所長 各位）
描画領域の不具合理由まで関数で指摘してくれるとは思わなかった。
今まで十二分に時間と経費をかけた評価でも この処理仕様が誤りであったなんて知りませんでした。 早急に直します。 論理的に説明して戴いて助かりました。 本当に評価出なんですか？
マスク時のビットマップがバンディング境界でくずれる潜在バグであるとのご指摘ありがとう。 さすがですね。お蔭ですぐに修正対応できました。
うち(設計サイド)の障害絞込みに要する工数が大幅にカットできました。 大変助かります。
障害用のアプリデータまで作って戴けて助かりました。 評価データ(アプリ)の見直しもお願いします。せめて作成ヒントだけでも戴けると助かります。
ひとりで数時間でROP全評価を完璧にできて しかも『これとこれ６箇所のパラメータ処理が違います』なんて言えないですよ。 あなた以外。
こんなツール(評価)があるなんて知らなかった。。。 用紙やインクのコストも大幅に減りますね。