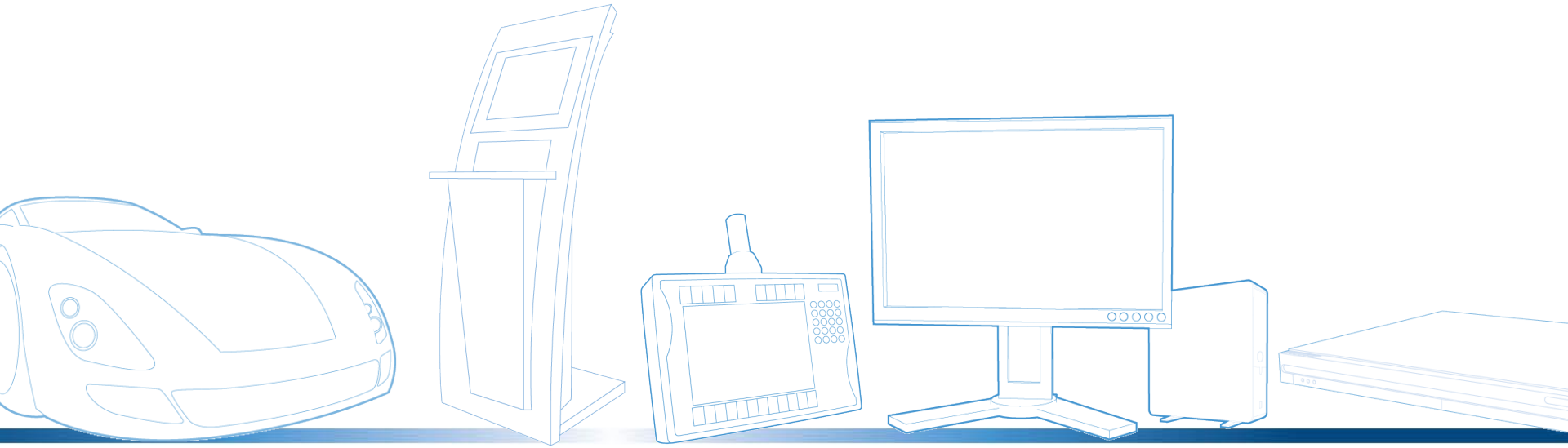
Windows Embedded
タッチパネルアプリケーションの開発

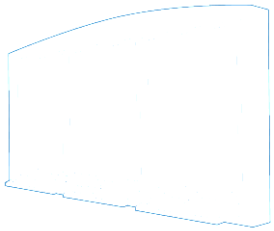
富士通ソフトウェアテクノロジーズ
杉本　拓也

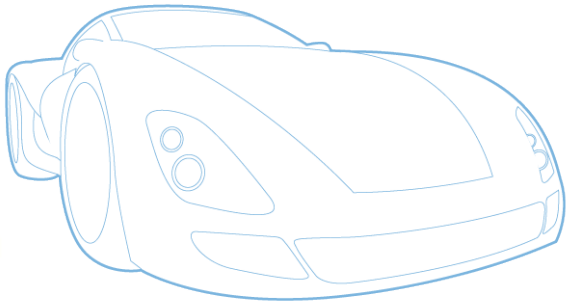


Windows Embedded

Agenda

- はじめに
- マルチタッチ
- ジェスチャー
- Windows Embedded Standard 7
- Windows Embedded Compact 7
- おまけ：Kinect
- まとめ

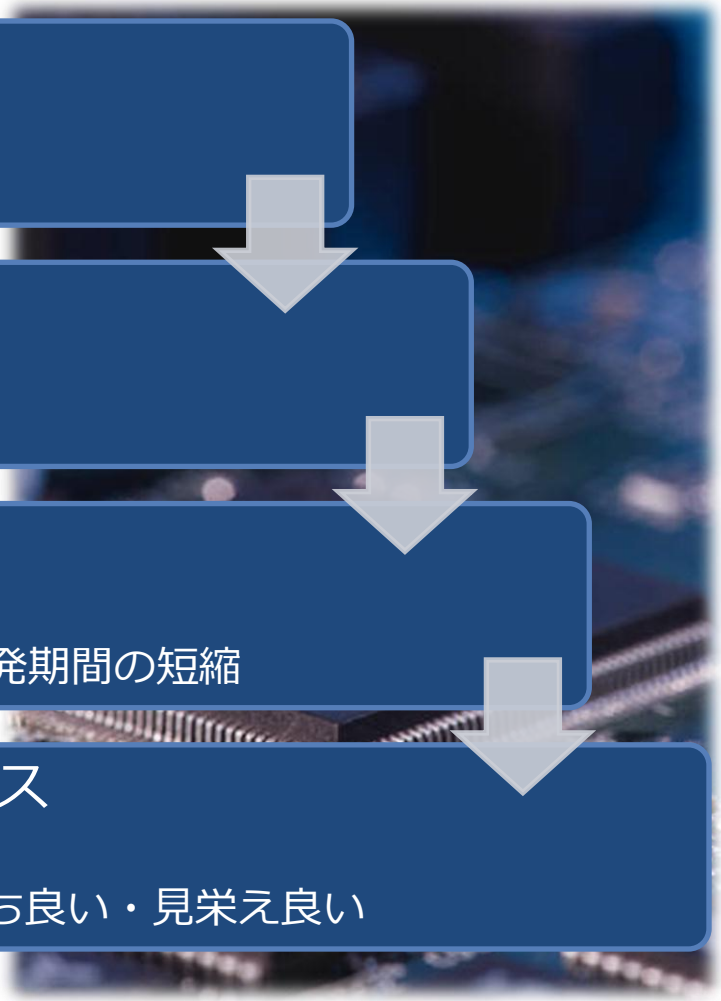




～はじめに～
組み込みデバイス開発の現状

Windows Embedded

組み込みデバイス開発を取り巻く状況



接続性

- クラウド・ネットワーク・他のデバイスとの接続

レガシーな機能

- リアルタイム性・省電力・安定性

開発コストの削減

- ソフトウェア・ハードウェアコスト・開発期間の短縮

新しいユーザインターフェース

- スマートフォンのようなUI・気持ち良い・見栄え良い

Windows® Embedded

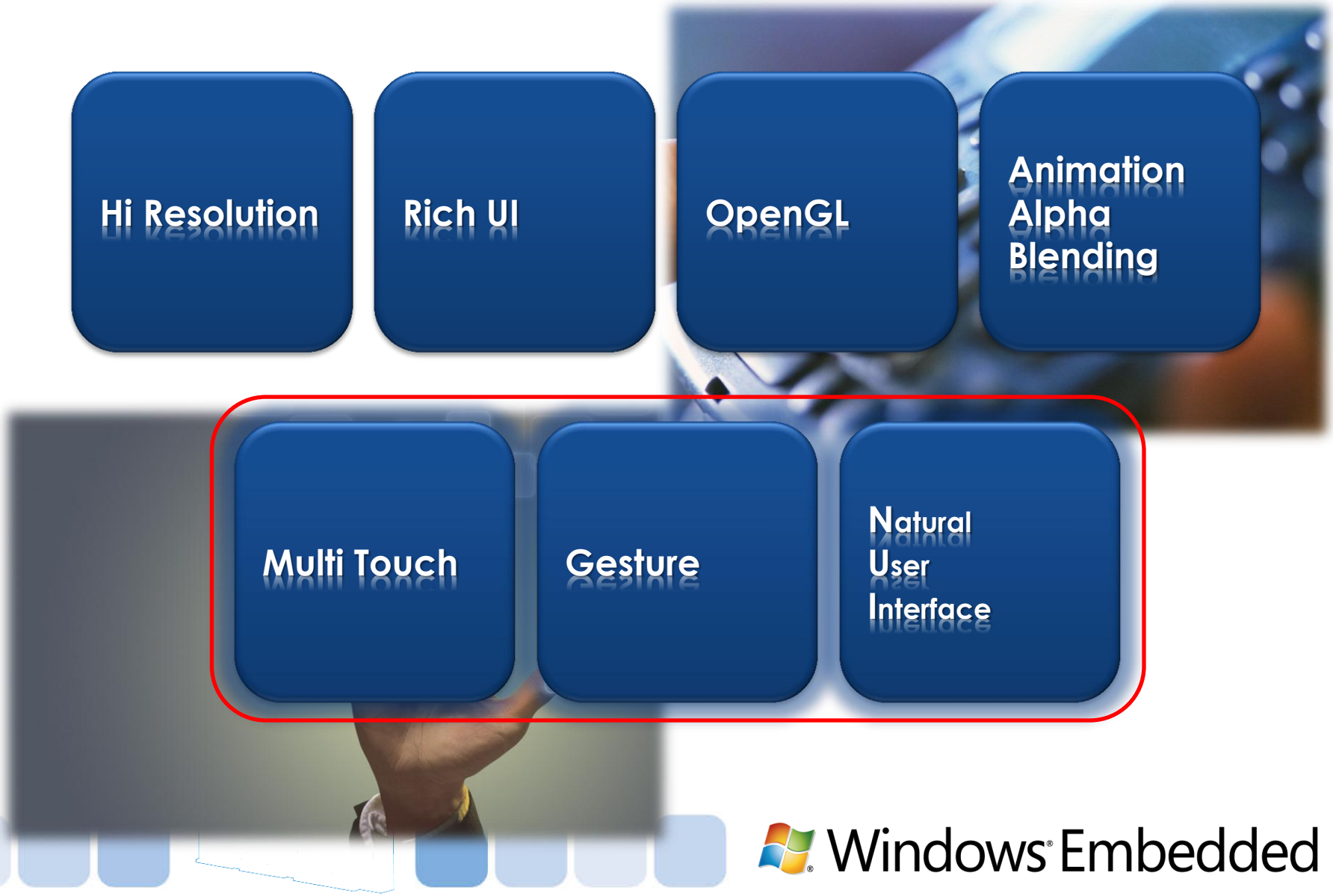
UI/UX

Windows 1.0 → Windows 7

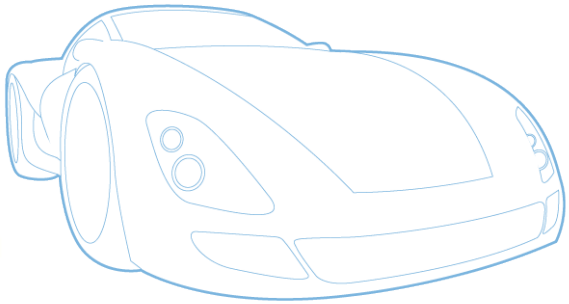
ファミコン → Wii

携帯電話 → スマートフォン

UI/UX



Windows Embedded

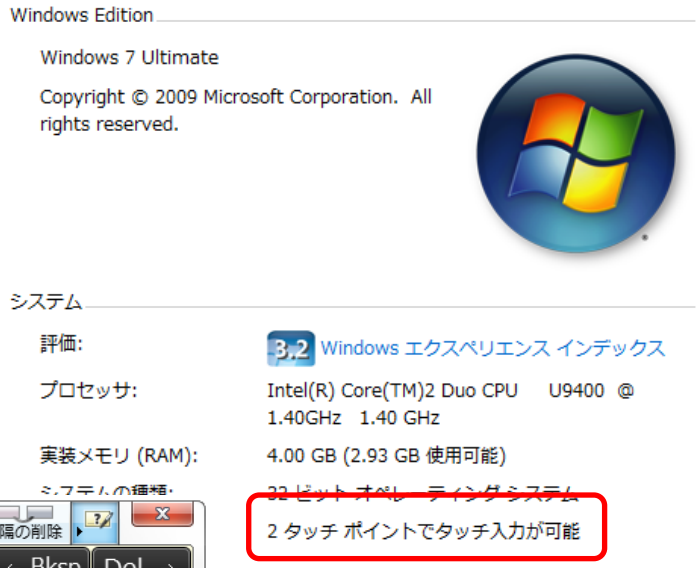


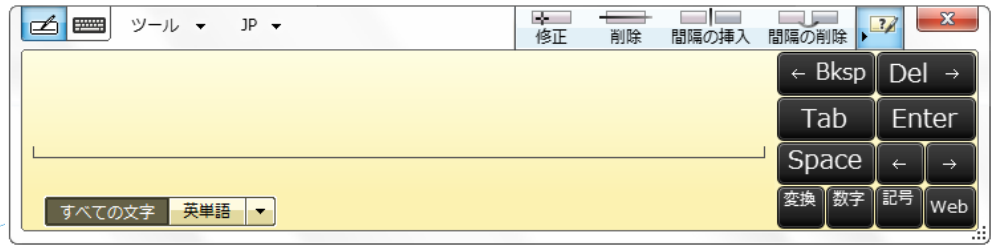
マルチタッチ機能

Windows Embedded

Windows 7のマルチタッチ機能

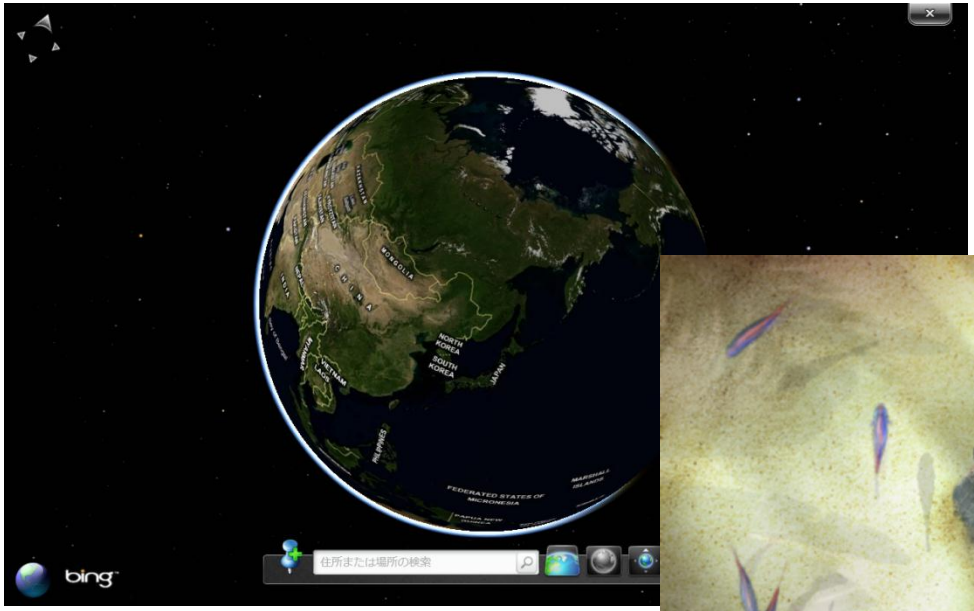
- タッチパネルが有効になると？
  - コンピュータのプロパティで「タッチパネル利用可能」
  - 入力パネル
  - IEの手のひらツール
  - IEのお気に入りの縦幅
  - Microsoft Officeの
    インク機能





マルチタッチ機能を満喫するなら

- Microsoft Touch Pack for Windows 7



Windows 8 Developer Preview

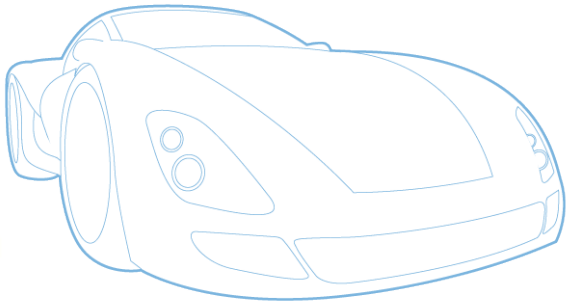
http://msdn.microsoft.com/en-us/windows/home/

Windows Embedded

Windows Phone 7

タッチパネル

- マトリクス・スイッチ
- 抵抗膜方式
- 表面弾性波方式
- 赤外線方式
- 電磁誘導方式
- 静電容量方式



マルチタッチアプリケーション開発

Windows Embedded Standard 7編

Windows Embedded

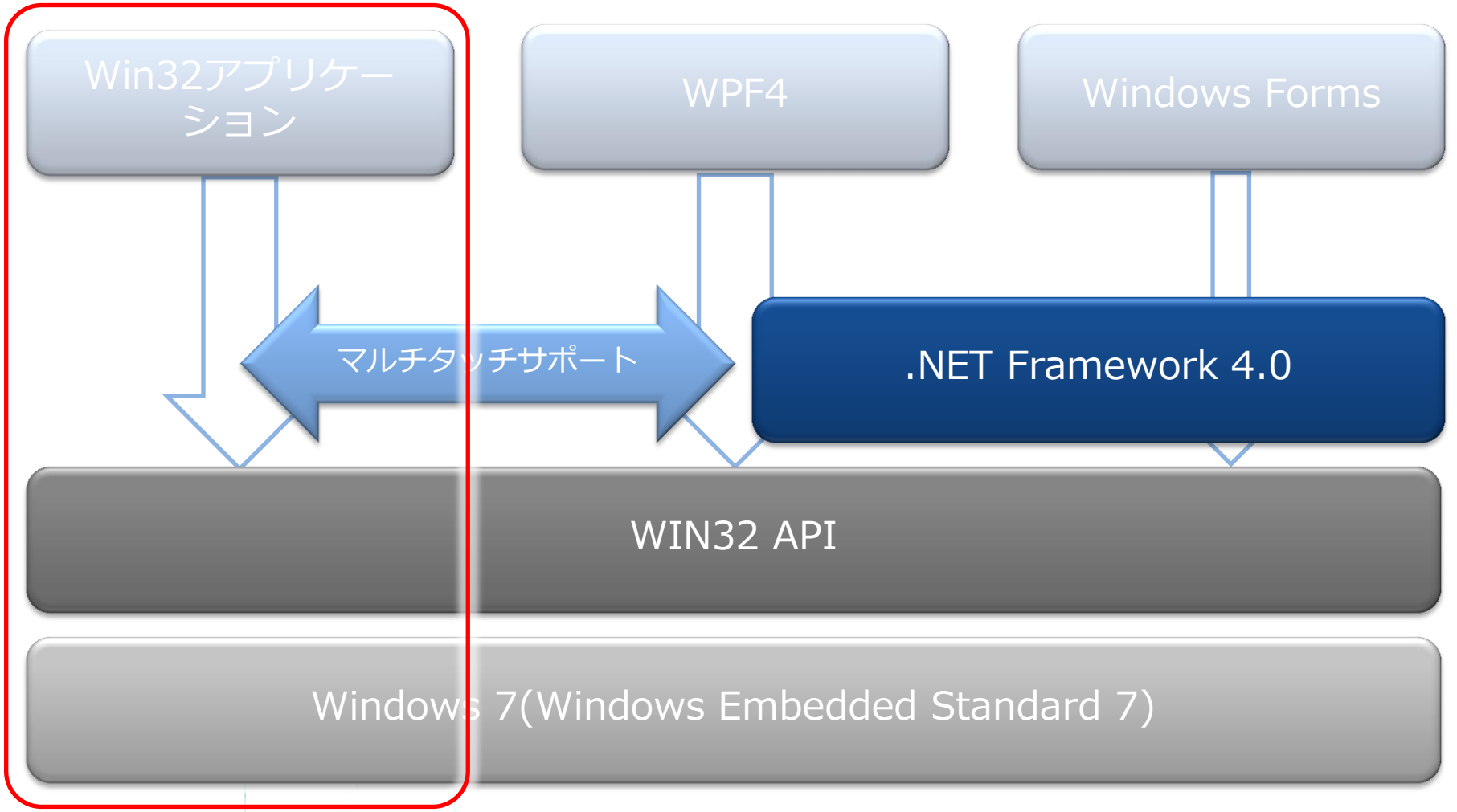
マルチタッチとジェスチャー

- レガシーサポート
  - Win32コントロールが標準的にサポート
  - 既存のマルチタッチ非対応アプリケーションでも操作可能
- マルチタッチ
  - ２本の指によるズーム・パン・フリック
- ジェスチャー
  - ズーム
  - 1 本または 2 本の指によるパン
  - 回転
  - 2 本の指によるタップ
  - プレス アンド タップ
  - フリック

ジェスチャーの種類

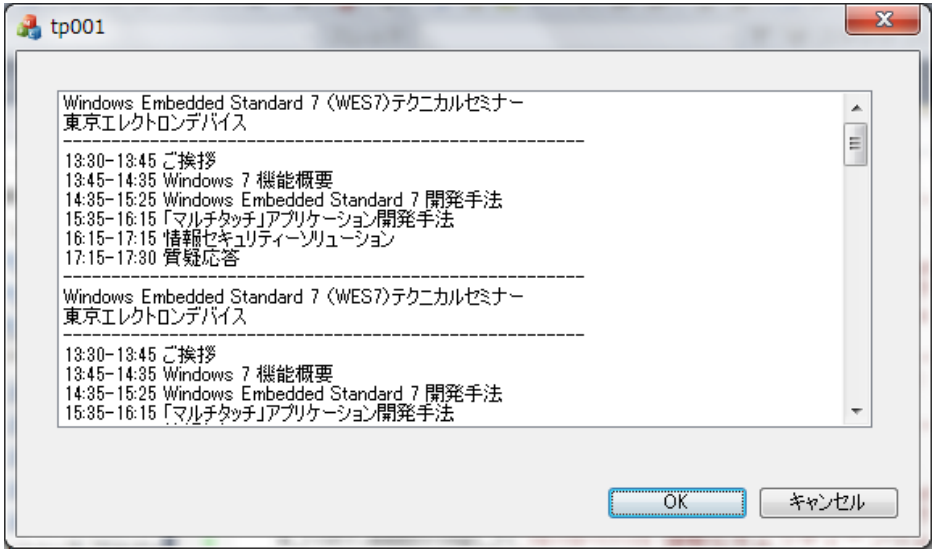
Windows Embedded

マルチタッチアプリケーションの動作環境



レガシーサポート

- DialogにListBoxを張り付ける
- コードは何も変更せずにフリックによる上下スクロールが可能
- 様々な標準コントロールがタッチ操作可能



Windows Embedded

マルチタッチを制御する

- Windows 7環境下でタッチ入力を制御するためには下記２つのメッセージを処理
- WM_TOUCH
  - タッチ入力の”生”データ
- WM_GESTURE
  - WM_TOUCHのデータが加工されジェスチャーデータとして受信可能

WM_GESTURE

- lParam
  - GESTUREINFOへのハンドル。GetGestureInfo()を利用して取得。
  - CloseGestureInfoHandle()でハンドルを閉じる
- GESTUREINFO
  - cbSize: 構造体のサイズ (バイト単位)。
  - ptsLocation: ジェスチャに関連付けられた座標が含まれる POINTS 構造体。
  - dwFlags: 開始、慣性、終了など、ジェスチャの状態。
  - ullArguments: 8 バイトに収まるジェスチャ引数を含む 64 ビットの符号なし整数。

SetGestureConfig

- GID_ROTATEは標準では送られてこない
- SetGestureConfigを利用して要求

```
GESTURECONFIG gestureConfig;

gestureConfig.dwID = GID_ROTATE;
gestureConfig.dwWant = GC_ROTATE;
gestureConfig.dwBlock = 0;
BOOL b = SetGestureConfig(hWnd, 0, 1, &gestureConfig,
sizeof(gestureConfig));
```

- すべてのジェスチャーコマンドが欲しいなら

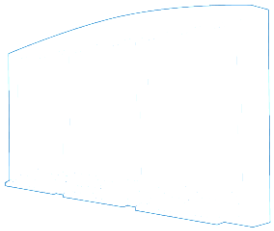
```
GESTURECONFIG gestureConfig;

gestureConfig.dwID = 0;
gestureConfig.dwBlock = 0;
gestureConfig.dwWant = GC_ALLGESTURES;

BOOL b = SetGestureConfig(hWnd, 0, 1, &gestureConfig,
sizeof(gestureConfig));
```

WM_GESTUREの処理

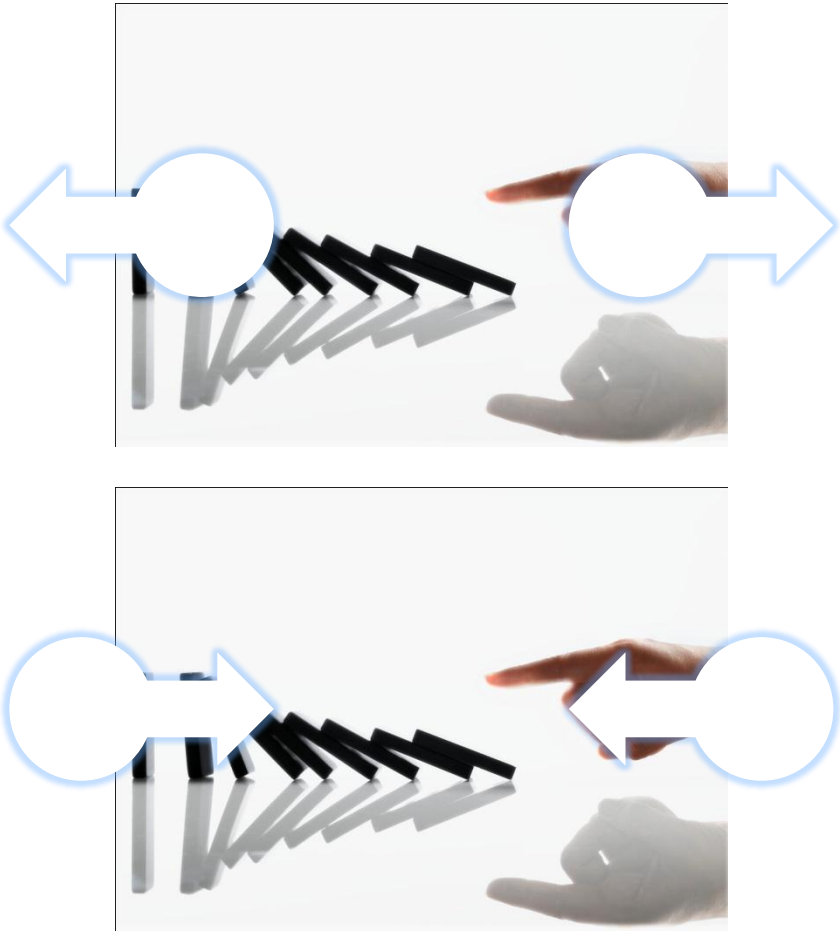
```
void CMTTestDlg::DecodeGesture(WPARAM wParam, LPARAM lParam)
{
    GESTUREINFO gi;
    ZeroMemory(&gi, sizeof(GESTUREINFO));
    GetGestureInfo((HGESTUREINFO)lParam, &gi);
    switch (gi.dwID){
        case GID_ZOOM:
            // Code for zooming goes here
            break;
        case GID_PAN:
            break;
        case GID_ROTATE:
            break;
        case GID_TWOFINGERTAP:
            break;
        case GID_PRESSANDTAP:
            break;
        default:
            // You have encountered an unknown gesture
            break;
    CloseGestureInfoHandle((HGESTUREINFO)lParam);
}
```



ズームジェスチャ

- GID_ZOOMを処理
- dwFlagsを解析

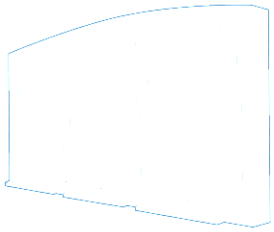
- GF_BEGIN: 最初のWM_GESTURE メッセージで受けとり、ジェスチャが開始されたことを示します。
- GF_INERTIA: ジェスチャで慣性操作がトリガーされたことを示します。
- GF_END: ジェスチャが終了したことを示します。



GID_ZOOMの処理

```
case GID_ZOOM:
switch(gi.dwFlags)
{
case GF_BEGIN:
    _dwArguments = LODWORD(gi.ullArguments);
    _ptFirst.x = gi.ptsLocation.x;
    _ptFirst.y = gi.ptsLocation.y;
    ScreenToClient(hWnd,&_ptFirst);
    break;
default:
    // We read here the second point of the gesture. This is middle point between fingers.
    _ptSecond.x = gi.ptsLocation.x;
    _ptSecond.y = gi.ptsLocation.y;
    ScreenToClient(hWnd,&_ptSecond);
    // We have to calculate zoom center point
    ptZoomCenter.x = (_ptFirst.x + _ptSecond.x)/2;
    ptZoomCenter.y = (_ptFirst.y + _ptSecond.y)/2;
    // The zoom factor is the ratio between the new and the old distance.
    k = (double)(LODWORD(gi.ullArguments))/(double)(_dwArguments);
    // Now we process zooming in/out of the object
    ProcessZoom(k,ptZoomCenter.x,ptZoomCenter.y);
    InvalidateRect(hWnd,NULL,TRUE);

    // Now we have to store new information as a starting information for the next step
    _ptFirst = _ptSecond;
    _dwArguments = LODWORD(gi.ullArguments);
    break;
}
break;
```

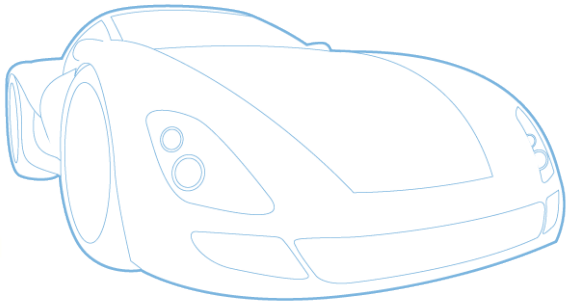


WM_TOUCH

- RegisterTouchWindow()を発行する
- wParam
  - ポイントの個数
- lParam
  - TOUCHINPUTへのハンドル。GetTouchInputInfoを利用して取得
  - CloseTouchInputHandleでハンドルを閉じる
- TOUCHINPUT
  - dwID: 特定のタッチ入力を他のタッチ入力と区別するタッチ ポイント ID。
  - dwFlags: タッチ ポイントの状態を示すビット フラグのセット。
  - X および Y: タッチ ポイントの X 座標と Y 座標
  - dwTime: イベントのタイム スタンプ (ミリ秒単位)。
  - dwMask: 構造体のオプション フィールドに有効な値が含まれているかどうかを示すビット フラグのセット。

WM_TOUCHの処理

```
case WM_TOUCH:
{
    unsigned int numInputs = (unsigned int) wParam;
    TOUCHINPUT* ti = new TOUCHINPUT[numInputs];
    if(GetTouchInputInfo((HTOUCHINPUT)lParam, numInputs, ti, sizeof(TOUCHINPUT)))
    {
        // Handle each contact point
        for(unsigned int i=0; i< numInputs; ++i)
        {
          /* handle ti[i]  */
        }
    }
    CloseTouchInputHandle((HTOUCHINPUT)lParam);
    delete [] ti;
}
break;
default:
    return DefWindowProc(hWnd, message, wParam, lParam);
```



マルチタッチアプリケーション開発

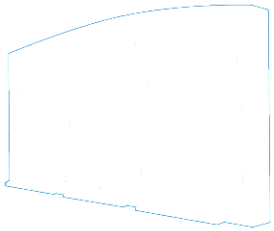
Windows Embedded Compact 7編

Windows Embedded

Windows Phone 7

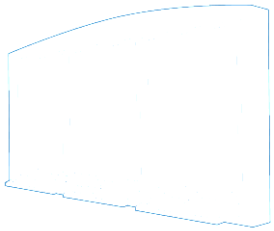
サポートされるジェスチャー

- 直接操作
- ダブルタップ
- フリック
- ホールド
- パン
- タップ



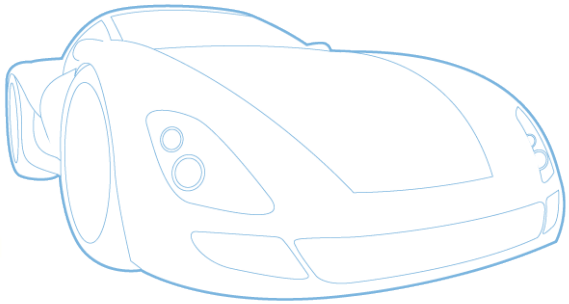
必要なSYGEN VARIABLES

Catalog Item	Sysgen Variable
Touch Gesture GWES component	SYSGEN_TOUCHGESTURE
Default Gesture Response	SYSGEN_GESTUREANIMATION
Gesture Animation Support	SYSGEN_PHYSICSENGINE



WM_GESTURE

- lParam
  - GESTUREINFOへのハンドル。GetGestureInfo()を利用して取得。
  - CloseGestureInfoHandle()でハンドルを閉じる
- GESTUREINFO
  - cbSize: 構造体のサイズ (バイト単位)。
  - ptsLocation: ジェスチャに関連付けられた座標が含まれる POINTS 構造体。
  - dwFlags: 開始、慣性、終了など、ジェスチャの状態。
  - ullArguments: 8 バイトに収まるジェスチャ引数を含む 64 ビットの符号なし整数。

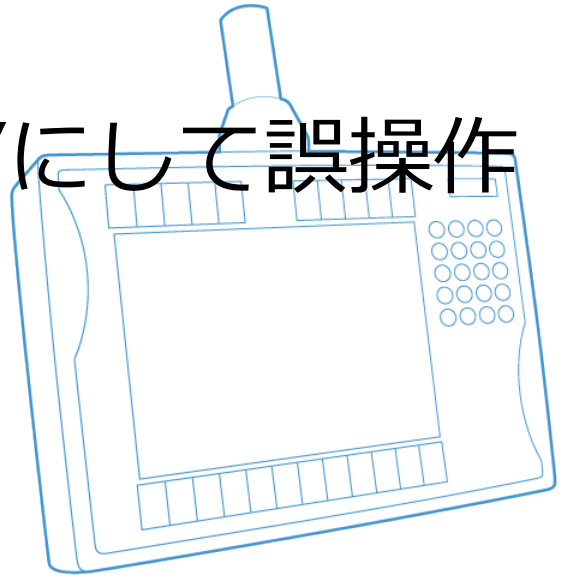


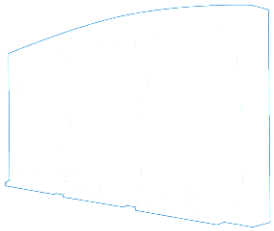
組み込みデバイスにマルチタッチ

Windows Embedded

タッチパネルデバイスの誤操作防止

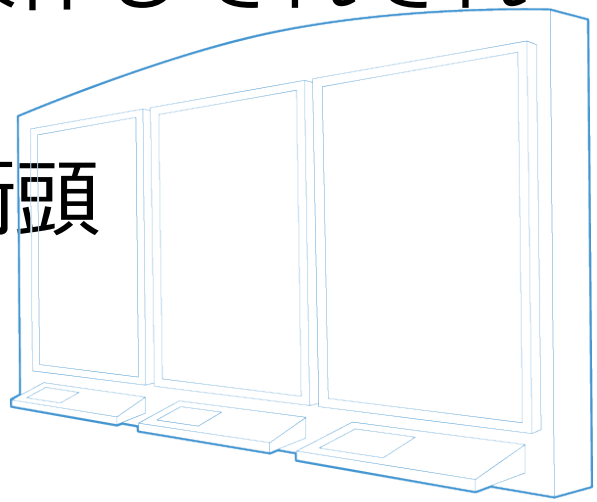
- 工場などの制御機にてマルチタッチを利用
- “安全ボタン”を押下しながら対象のボタンを押下
- ユーザの操作を“あえて複雑”にして誤操作を防ぐ

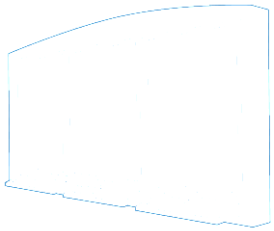


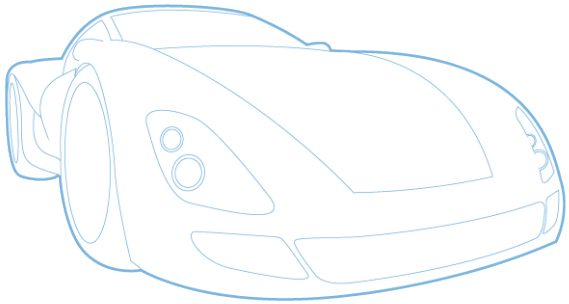


複数人で入力

- デジタルサイネージなど複数の人が操作する可能性があるデバイスでマルチタッチを利用
- 大画面で複数のユーザが操作しそれぞれ異なる情報を表示
- 文教・オフィス・店頭・街頭





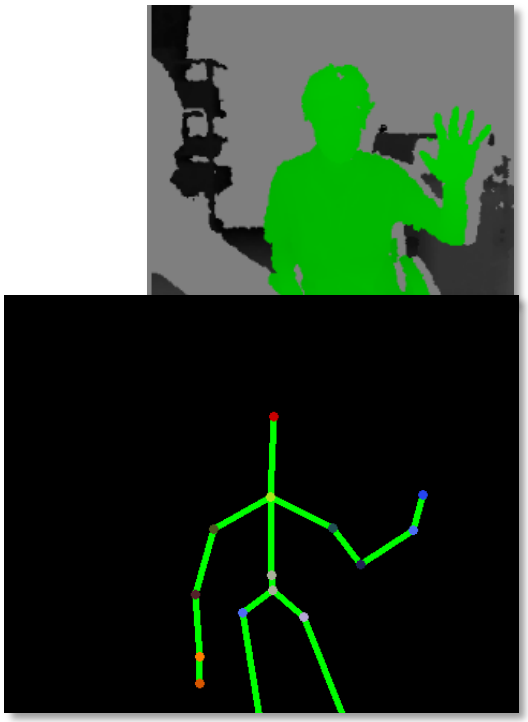


おまけ：
Kinect for Windows SDK

Windows Embedded

Kinect for Windows SDK

- Windows 7からKinectデバイスを利用可能とするSDK
- 開発環境
  - Visual Studio® 2010
  - Kinect for Windows SDK Beta
  - Direct X® SDK
  - Speech Platform SDK
  - C++ / C# / VB



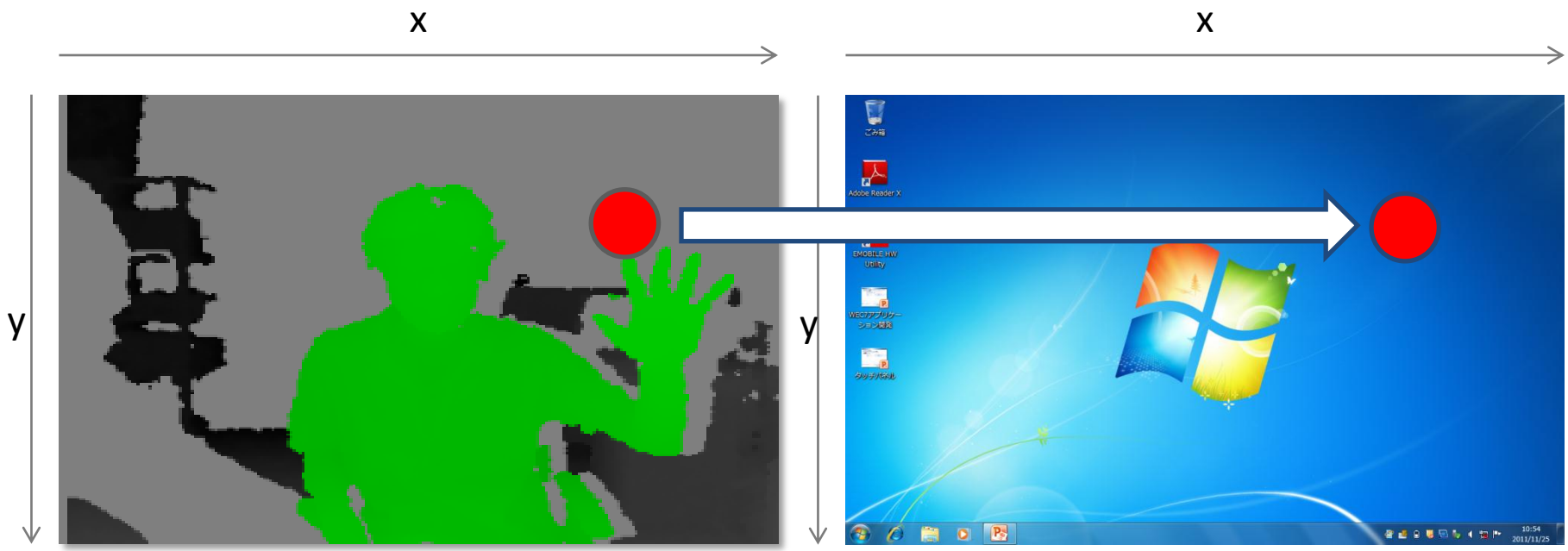
Kinect for Windows SDK Beta2

- http://www.kinectforwindows.org

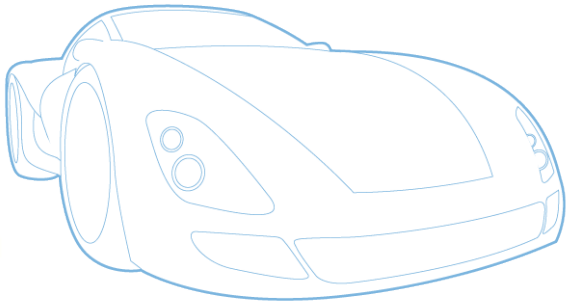
- スケルトン追跡の高速化
- 関節追跡の精度向上
- Kinectの挿抜検出

- PC向けのKinectハード
  - USBケーブル短め
  - 近い画像認識の強化

タッチパネルとKinect



Kinectで検出されるポイントをスクリーンと対比



まとめ

Windows Embedded

タッチパネルを利用することで

- マルチタッチ
  - タッチパネル操作の誤操作防止
- ジェスチャー
  - OS標準以外のジェスチャーもWM_TOUCHを制御することで実装可能

- WPF・Silverlightなど最新のプラットフォームを利用しなくてもマルチタッチ対応は可能
  - 既存のXpe/WESのアプリケーションもタッチ操作に！