Word for UWP

2018.04.10 更新

グレープシティ株式会社

Word for UWP

目次

Word for UWP の概要

ComponentOne に導入された **Word for UWP** には、高度な機能を備えた Word ドキュメントを作成するために機能豊富な API が提供されています。**Word for UWP** は、Microsoft Word Open XML 形式の拡張子（*.docx）を持つ Word ドキュメントおよびリッチテキスト形式の拡張子（*.rtf）を持つ RTF ドキュメントの作成、読み取り、書き込みを行うことができます。

Word for UWP は、**C1WordDocument** クラス（C1.UWP.Word アセンブリのメンバ）を使用します。このクラスは、Microsoft Word ドキュメントと RTF ドキュメントの生成に必要な高度なプロパティおよびメソッドをすべて提供します。**Word for UWP** を使用して生成されたドキュメントは、ファイルシステムに保存することも、標準の Microsoft Word ドキュメント形式にエクスポートすることも簡単です。

主な特長

Word for UWP の主要な機能は次のとおりです。

- **充実したオブジェクトモデル**
  Word for UWP は、機能豊富で強力だが簡単にプログラム可能なオブジェクトモデルを提供します。高度なプロパティとメソッドをすべて提供する **C1WordDocument** クラスだけを使用して、Microsoft Word および RTF ドキュメントの両方を作成できます。

- **高度なライブラリ**
  Word for UWP は、高度なメソッドとして、Word ドキュメント内に画像、テキスト、グラフィック、引用文、および表を追加します。

- **描画と再生**
  Word for UWPを使用して Word ドキュメント内で Windows Runtime User Interface (UI) オブジェクト (FrameworkElements) を描画および再生します。

- **さまざまな用紙サイズ**
  Word for UWP を使用して、さまざまな用紙サイズと向きでドキュメントを作成します。

- **テキストの描画**
  Word for UWP を使用すると、Word ドキュメント内でさまざまなフォントでテキストを描画したり、フォントプロパティを使用することができます。

- **表の追加**
  Word for UWP の RTFTable オブジェクトを使用して、表セルにデータを追加できます。

- **目次の追加**
  Word for UWP では、テキストと見出しを含むドキュメントに目次 (TOC) を追加できます。目次を使用して、ドキュメント内の別のページに移動することもできます。

- **ハイパーリンクとブックマーク**
  Word for UWP は、ドキュメント内を移動するためのブックマークと、別の URL に移動するためのハイパーリンクを提供します。

- **テキストフロー**
  Word for UWP を使用すると、Word ドキュメント内の段やページにテキストをフロー挿入できます。

オブジェクトモデルの概要

Word for UWP は、機能豊富で強力だが簡単にプログラム可能なオブジェクトモデルを提供します。C1WordDocument クラスは、Microsoft Word および RTF ドキュメントを作成するための高度なプロパティとメソッドをすべて提供します。

C1WordDocument Object
Add, **AddBookmark**, **AddBookmarkStart**, **AddBookmarkEnd**, **AddLink**, **AddParagraph**, **AddPicture**, **ColumnBreak**, **DrawArc**, **DrawRectangle**, **DrawString**, **FillPie**, **LoadAsync**, **Count**, **Current**, **Hyperlink**, **ShapesWord2007Compatible**
Font
Bold, **FontFamily**, **Italic**, **Name**, **Size**, **Strikeout**, **Style**, **Underline**
Pen
Color, **DashPattern**, **DashStyle**
RTFPageSize Object
A0, **A1**, **A4**, **A10**, **B1**, **B5**, **HalfLetter**, **Legal**, **Letter**
StringFormat
Alignment, **Angle**, **LineAlignment**, **LineSpacing**

クイックスタート

クイックスタートガイドでは、**Word for UWP**について詳しく説明します。このセクションでは、Visual Studio で新しい UWP プロジェクトを作成する方法を学びます。ドキュメントを作成して保存するには、**C1Word** の参照 (dll) とボタンをアプリケーションに追加する必要があります。

手順 1:アプリケーションの設定

最初に Visual Studio で UWP アプリケーションを作成し、次にそのアプリケーションに次の手順で **C1Word** 参照とボタンコントロールを追加します。

1. Visual Studio で新しい UWP プロジェクトを作成します。
2. アプリケーションに **C1Word** 参照(dll)を追加します。
3. コードビューで、次の名前空間を追加します。

Visual Basic

```
Imports C1.Xaml.Word
Imports C1.Xaml.Word.Objects
```

C#

```
using C1.Xaml.Word;
using C1.Xaml.Word.Objects;
```

4. デザインビューに切り替えて、アプリケーション内のフォームに Button コントロールを追加し、**Word for UWP** の使用を開始します。**Content** プロパティに **Text** などの適切なテキストを設定し、**Name** プロパティを **btnText** に設定し、**Click** イベントを **btnText_Click** に設定します。
5. プロパティウィンドウで **btnText_Click** イベントをダブルクリックします。
コードビューに **btnText_Click** イベントが作成されます。

手順 2:テキストの追加

Visual Studio プロジェクトでコードビューを表示したまま、**btnText_Click** イベントに次のコードを追加します。

Visual Basic

```
Dim word As New C1WordDocument()

' 横方向を使用して効果を高めます
word.Landscape = True

' テキストを測定および表示します
Dim text = "こんにちは!! これはサンプルテキストです。"
Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)

' 段落を追加します
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify)
```

```
Dim picker As New FileSavePicker()
picker.FileTypeChoices.Add("Word Open XML Format (.docx)", New List(Of
String)() From { _
        ".docx" _
})
picker.FileTypeChoices.Add("RTF Format (.rtf)", New List(Of String)()
From { _
        ".rtf" _
})

picker.DefaultFileExtension = ".docx"
picker.SuggestedStartLocation = PickerLocationId.DocumentsLibrary
Dim file As StorageFile = Await picker.PickSaveFileAsync()
If file IsNot Nothing Then
        Await word.SaveAsync(file,
If(file.Name.ToLower().EndsWith(".docx"), FileFormat.OpenXml,
FileFormat.Rtf))

        Dim dlg As New MessageDialog("ファイルを保存しました")
        Await dlg.ShowAsync()
End If
```

C#

```
C1WordDocument word = new C1WordDocument();

// 横方向を使用して効果を高めます
word.Landscape = true;

// テキストを測定および表示します
var text = "こんにちは!! これはサンプルテキストです。";
var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);

// 段落を追加します
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify);

FileSavePicker picker = new FileSavePicker();
picker.FileTypeChoices.Add("Word Open XML Format (.docx)", new List <
string > () {
  ".docx"
});
picker.FileTypeChoices.Add("RTF Format (.rtf)", new List < string > () {
  ".rtf"
});

picker.DefaultFileExtension = ".docx";
picker.SuggestedStartLocation = PickerLocationId.DocumentsLibrary;
StorageFile file = await picker.PickSaveFileAsync();
if (file != null) {
  await word.SaveAsync(file, file.Name.ToLower().EndsWith(".docx") ?
FileFormat.OpenXml : FileFormat.Rtf);
```

```
    MessageDialog dlg = new MessageDialog("ファイルを保存しました");
    await dlg.ShowAsync();
}
```

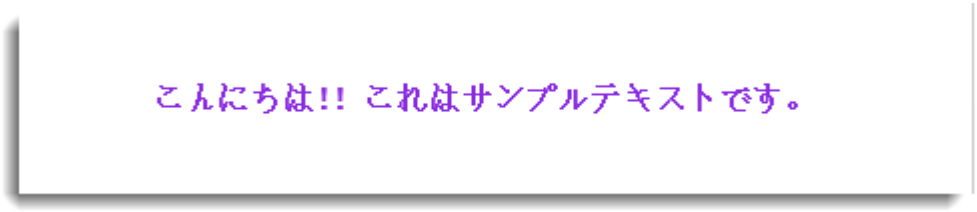
上記のコードでは、Word ドキュメントが横長モードで作成され、AddParagraph メソッドを使用してテキストが追加されます。作成したドキュメントは、選択した場所に保存できます。目的の場所を選択し、ドキュメントの名前を書き込むこともできます。ドキュメントは、その名前で保存されます。

手順 3：アプリケーションの実行

前の手順では、テキストを作成して追加し、Word ドキュメントを保存するコードを **btnText_Click** イベントに追加しました。この手順では、アプリケーションを実行して、作成したドキュメントを表示します。次の手順に従って、アプリケーションを実行します。

1. **[F5]**キーを押してアプリケーションを実行します。
2. ボタンをクリックして、**Word for UWP** を使用して作成および保存したドキュメントを表示します。

ドキュメントが開かれると、次の図のように表示されます。



Word for UWP の操作

Word for UWP は機能豊富な API とオブジェクトモデルを備えており、Word ドキュメントのほかに、Microsoft Word や他のエディタでサポートされている RTF ドキュメントも作成できます。**Word for UWP** の詳細な仕組みを次のトピックに示します。

基礎レベルの操作

Word for UWP を使用して、画像、グラフィック、引用文などの単純なイラストを Word ドキュメントに追加できます。**Word for UWP** を使用すると、このようなイラストを数行のコードで追加できます。次のトピックでは、単純なテキストといくつかの基本的なイラストを追加する方法について説明します。

テキストの追加

Word for UWP を使用して、Word ドキュメントにテキストを追加できます。目的のテキストを記述し、**AddParagraph** メソッドを使用してそのテキストを追加する必要があります。Word ドキュメントに表示するテキストに対して、Font クラスとそのプロパティを使用して、フォントのスタイル、ファミリ、色などのプロパティを設定することもできます。Word ドキュメントへのテキストの追加は、次のコードで実装されます。

1. 次のコードで、**C1WordDocument** クラスのオブジェクトを作成します。

Visual Basic

```
Dim word As New C1WordDocument()
```

C#

```
C1WordDocument word = new C1WordDocument();
```

2. ドキュメントにテキストを追加するには、次のコードを記述します。

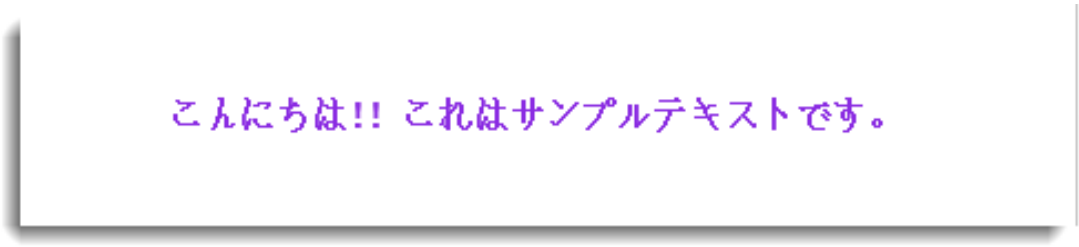
Visual Basic

```
Dim text = "こんにちは!! これはサンプルテキストです。"
Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify)
WordUtils.Save(word)
```

C#

```
var text = "こんにちは!! これはサンプルテキストです。";
var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify);
WordUtils.Save(word);
```

ドキュメントは、次の図のように表示されます。



画像の追加

Word ドキュメントにテキストに加えて画像を挿入して、全体的に見栄えをよくしたい場合があります。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\UWP\WordSamples
これらのクラスを上記の場所からアプリケーションで使用できます。

ドキュメントに画像を追加するには、次のコードを使用します。これは、画像をロードしてドキュメント内にスケッチします。

Visual Basic

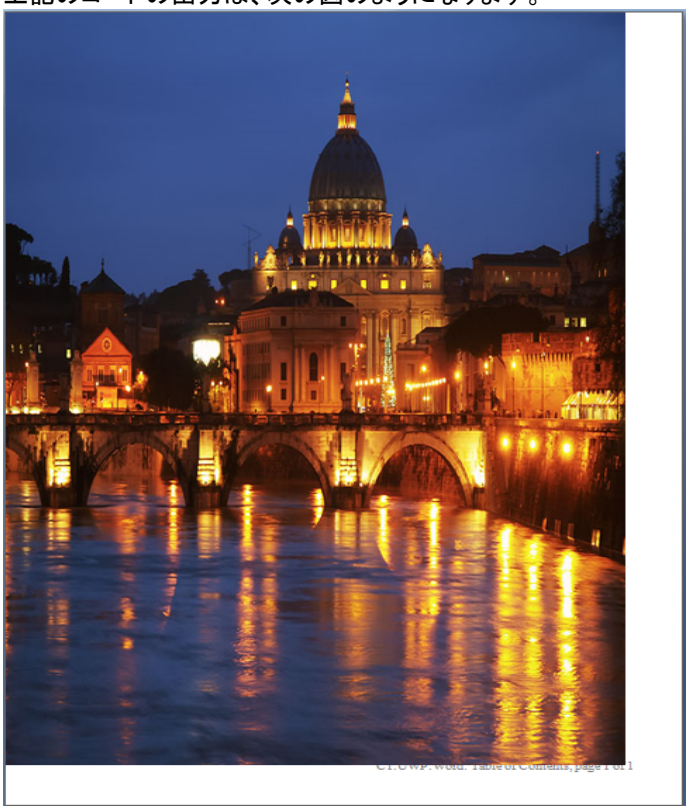
```
' ページ四角形を計算します(マージンを差し引いて)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim ras As New InMemoryRandomAccessStream()
' 書き込み可能なビットマップに画像をロードします
Dim wb As New WriteableBitmap(880, 660)
Dim file = Await StorageFile.GetFileFromApplicationUriAsync(New Uri("ms-
appx:///WordSamplesLib/Assets/pic.jpg"))

wb.SetSource(Await file.OpenReadAsync())
Dim rcPic As New Rect(New Point(0, 0), New Point(word.PageSize.Width,
word.PageSize.Height))

' アスペクト比を維持してページ上に描画します
word.DrawImage(wb, rcPic)
WordUtils.Save(word)
```

C#

```
// ページ四角形を計算します(マージンを差し引いて)
Rect rcPage = WordUtils.PageRectangle(word);
InMemoryRandomAccessStream ras = new InMemoryRandomAccessStream();
// 書き込み可能なビットマップに画像をロードします
WriteableBitmap wb = new WriteableBitmap(880, 660);
var file = await StorageFile.GetFileFromApplicationUriAsync(new Uri("ms-
appx:///WordSamplesLib/Assets/pic.jpg"));

wb.SetSource(await file.OpenReadAsync());
Rect rcPic = new Rect(new Point(0, 0), new Point(word.PageSize.Width,
word.PageSize.Height));

// アスペクト比を維持してページ上に描画します
word.DrawImage(wb, rcPic);
WordUtils.Save(word);
```

上記のコードで、画像は書き込み可能なビットマップにロードされ、DrawImage メソッドを使用して描画されます。

上記のコードの出力は、次の図のようになります。



グラフィックの描画

グラフィックを追加することで、ドキュメントの見栄えがよくなり、視覚に訴えることができます。ドキュメントに、円弧、ベジェ、楕円、直線、円、多角形、折れ線、四角形などのさまざまなタイプの図形を追加できます。テキストに加えて円、四角形、ポリライン、ベジェなどのグラフィックを追加するには、次のコードを使用します。

Visual Basic

```
Dim word = New C1WordDocument()
' 描画の設定を行います
Dim rc As New Rect(100, 100, 300, 200)
Dim text As String = "Hello world of .NET Graphics and Word/RTF." & vbCr
& vbLf & "よろしくお願いします。"
Dim font As New Font("Times New Roman", 12, RtfFontStyle.Italic Or
RtfFontStyle.Underline)

' ワードドキュメントに描画します
Dim penWidth As Integer = 0
Dim penRGB As Byte = 0
word.FillPie(Colors.DarkRed, rc, 0, 20F)
word.FillPie(Colors.Green, rc, 20F, 30F)
word.FillPie(Colors.Teal, rc, 60F, 12F)
word.FillPie(Colors.Orange, rc, -80F, -20F)
For startAngle As Single = 0 To 359 Step 40
        Dim penColor As Color = Color.FromArgb(&Hff, penRGB, penRGB,
penRGB)
```

```
        Dim pen As New Pen(penColor,
System.Math.Max(System.Threading.Interlocked.Increment(penWidth),penWidth
- 1))
        penRGB = CByte(penRGB + 20)
        word.DrawArc(pen, rc, startAngle, 40F)
Next
word.DrawRectangle(Colors.Red, rc)


' ベジェ曲線を表示します
Dim pts = New Point() {New Point(400, 100), New Point(420, 130), New
Point(500, 140), New Point(530, 120)}

' ベジェを描画します
word.DrawBeziers(New Pen(Colors.Green, 4), pts)


' ベジェ制御点を表示します
word.DrawPolyline(Colors.Gray, pts)
For Each pt As Point In pts
        word.FillRectangle(Colors.Orange, pt.X - 2, pt.Y - 2, 4, 4)
Next

' タイトル
word.DrawString(Strings.Bezier, font, Colors.Black, New Rect(500, 150,
100, 100))
```

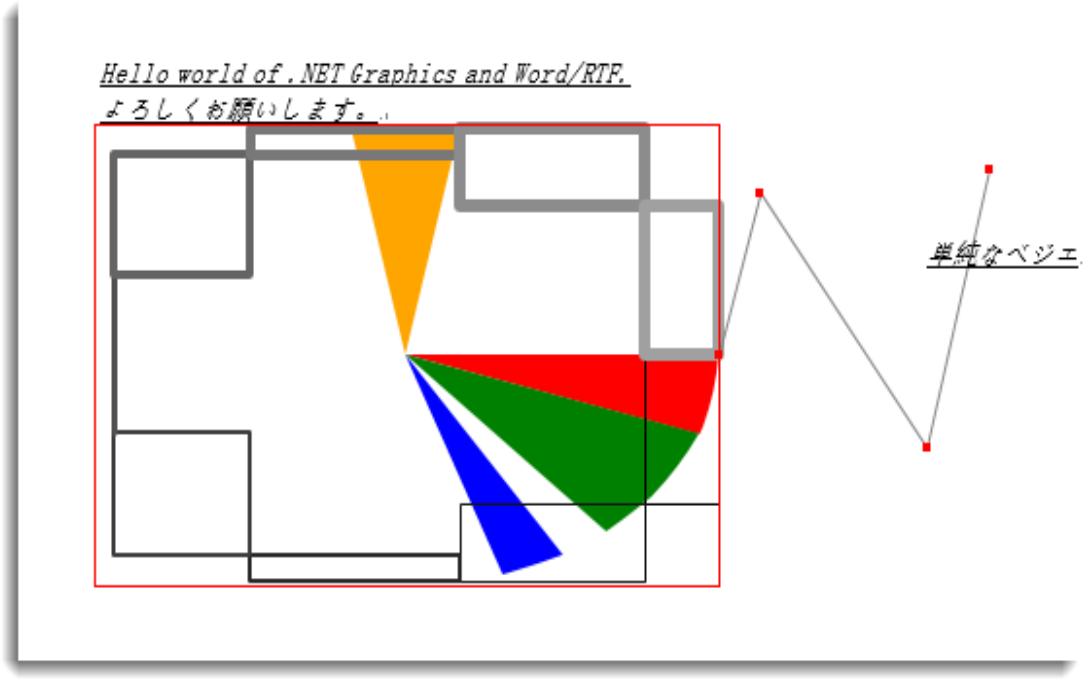
C#

```
var word = new C1WordDocument();
// 描画の設定を行います
Rect rc = new Rect(100, 100, 300, 200);
string text = "Hello world of .NET Graphics and Word/RTF.\r\nよろしくお願いします。";
Font font = new Font("Times New Roman", 12, RtfFontStyle.Italic |
RtfFontStyle.Underline);

// ワードドキュメントに描画します
int penWidth = 0;
byte penRGB = 0;
word.FillPie(Colors.DarkRed, rc, 0, 20 f);
word.FillPie(Colors.Green, rc, 20 f, 30 f);
word.FillPie(Colors.Teal, rc, 60 f, 12 f);
word.FillPie(Colors.Orange, rc, -80 f, -20 f);
for (float startAngle = 0; startAngle < 360; startAngle += 40) {
  Color penColor = Color.FromArgb(0xff, penRGB, penRGB, penRGB);
  Pen pen = new Pen(penColor, penWidth++);
  penRGB = (byte)(penRGB + 20);
  word.DrawArc(pen, rc, startAngle, 40 f);
}
word.DrawRectangle(Colors.Red, rc);
```

```
// ベジェ曲線を表示します
var pts = new Point[] {
  new Point(400, 100), new Point(420, 130),
    new Point(500, 140), new Point(530, 120),
};

// ベジェを描画します
word.DrawBeziers(new Pen(Colors.Green, 4), pts);


// ベジェ制御点を表示します
word.DrawPolyline(Colors.Gray, pts);
foreach(Point pt in pts) {
  word.FillRectangle(Colors.Orange, pt.X - 2, pt.Y - 2, 4, 4);
}

// タイトル
word.DrawString(Strings.Bezier, font, Colors.Black, new Rect(500, 150,
100, 100));
```

上記のコードでは、**DrawRectangle**、**DrawArc**、**DrawPolyline**、および **DrawBeziers** メソッドを使用してさまざまなタイプのグラフィック（直線、四角形、円、ベジェなど）を描画しています。また、**DrawString** メソッドを使用してテキストを描画してドキュメントに表示しています。

上記のコードの出力は、次の図のようになります。



引用文の追加

Word for UWP を使用して、別のファイルからの引用文をドキュメントに追加できます。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。

Documents\ComponentOne Samples\UWP\WordSample
これらのクラスを上記の場所からアプリケーションで使用できます。

次のコードを使用して、テキストファイルからの引用文をドキュメントに追加します。

Visual Basic

```
' ページ四角形を計算します(マージンを差し引いて)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim rc As Rect = rcPage

' 出力パラメータを初期化します
Dim hdrFont As New Font("Arial", 14, RtfFontStyle.Bold)
Dim titleFont As New Font("Arial", 24, RtfFontStyle.Bold)
Dim txtFont As New Font("Times New Roman", 10, RtfFontStyle.Italic)

' タイトルを追加します
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rc)
' ドキュメントを構築します
For Each s As String In GetQuotes()
        Dim authorQuote As String() = s.Split(ControlChars.Tab)

        ' ヘッダーをレンダリングします(作成者)
        Dim author = authorQuote(0)
        rc.Y += 20
        rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage,
rc, True)

        ' 本文をレンダリングします(引用文)
        Dim text As String = authorQuote(1)
        rc.X = rcPage.X + 36
        ' << 本文を 1/2 インチインデントします
        rc.Width = rcPage.Width - 40
        rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc)
        rc.X = rcPage.X
        ' << インデントを元に戻します
        rc.Width = rcPage.Width ' << 各引用文の後に 12pt のスペースを追加します
        rc.Y += 12
Next
Private Shared Function GetQuotes() As List
        Dim list = New List()

Using sr = New StreamReader(GetType(BasicTextPage).GetTypeInfo()

.Assembly.GetManifestResourceStream("WordSamples.Resources.quotes.txt"))
                Dim quotes = sr.ReadToEnd()
                For Each quote As String In quotes.Split("*"C)
                        Dim pos As Integer = quote.IndexOf(vbCr & vbLf)
                        If pos > -1 Then
                                Dim q = String.Format("{0}" & vbTab & "
{1}", quote.Substring(0, pos), quote.Substring(pos + 2).Trim())
                                list.Add(q)
```

```
                        End If
                Next
        End Using

        Return list
End Function
```

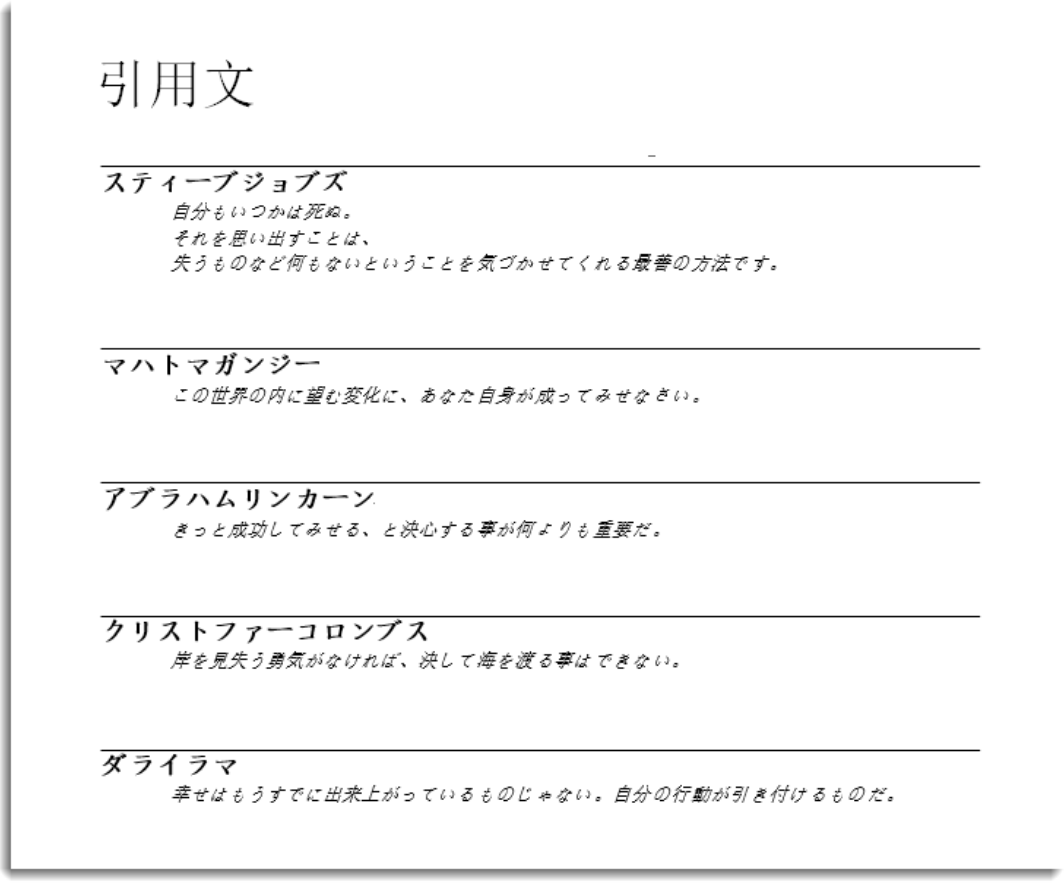
C#

```
// ページ四角形を計算します(マージンを差し引いて)
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = rcPage;

// 出力パラメータを初期化します
Font hdrFont = new Font("Arial", 14, RtfFontStyle.Bold);
Font titleFont = new Font("Arial", 24, RtfFontStyle.Bold);
Font txtFont = new Font("Times New Roman", 10, RtfFontStyle.Italic);

// タイトルを追加します
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rc);

// ドキュメントを構築します
foreach(string s in GetQuotes()) {
  string[] authorQuote = s.Split('\t');

  // ヘッダーをレンダリングします(作成者)
  var author = authorQuote[0];
  rc.Y += 20;
  rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage, rc,
true);

  // 本文をレンダリングします(引用文)
  string text = authorQuote[1];
  rc.X = rcPage.X + 36; // << 本文を 1/2 インチインデントします
  rc.Width = rcPage.Width - 40;
  rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc);
  rc.X = rcPage.X; // << インデントを元に戻します
  rc.Width = rcPage.Width;
  rc.Y += 12; // << 各引用文の後に 12pt のスペースを追加します
}

static List GetQuotes() {
  var list = new List();

using(var sr = new StreamReader(typeof(BasicTextPage).GetTypeInfo()

.Assembly.GetManifestResourceStream("WordSamples.Resources.quotes.txt")))
{
    var quotes = sr.ReadToEnd();
    foreach(string quote in quotes.Split('*')) {
      int pos = quote.IndexOf("\r\n");
      if (pos > -1) {
```

```
            var q = string.Format("{0}\t{1}", quote.Substring(0, pos),
quote.Substring(pos + 2).Trim());
            list.Add(q);
        }
    }
  }

  return list;
}
```

上記のコードは、テキストファイルから引用文を読み込んでドキュメントに書き出します。まずタイトルをドキュメントに追加し、次にヘッダーと本文をレンダリングしてから、ドキュメントにテキストを書き込みます。

上記のコードの出力は、次の図のようになります。



上級レベルの操作

通常、Word ドキュメントにはテキストが入りますが、これに画像、イラスト、表、メタファイルなどを追加することで、見栄えをよくし、かつわかりやすくすることができます。次に示すトピックでは、**Word for UWP** を使用して、これらの高度な機能を Word ドキュメントに追加する方法を説明します。

表の挿入

表は、Word ドキュメントでデータをいくつかの行と列に整えて表示するために使用されます。**Word for UWP** を使用すると、次のコードで表を Word ドキュメントに追加できます。

Visual Basic

```
' 表を追加します
word.LineBreak()
Dim rows As Integer = 4
Dim cols As Integer = 2
Dim table As New RtfTable(rows, cols)
word.Add(table)
table.Rows(0).Cells(0).SetMerged(1, 2)

For row As Integer = 0 To rows - 1
        If row = 0 Then
                table.Rows(row).Height = 50
        End If
        For col As Integer = 0 To cols - 1
                If row = 0 AndAlso col = 0 Then
                        text = Strings.DocumentBasicText2
                        table.Rows(row).Cells(col).Alignment =
ContentAlignment.MiddleCenter
                        table.Rows(row).Cells(col).BackFilling =
Colors.LightPink
                Else
                        text = String.Format("表のセル{0}:{1}", row, col)
                        table.Rows(row).Cells(col).BackFilling =
Colors.LightYellow
                End If
                table.Rows(row).Cells(col).Content.Add(New
RtfString(text))
                table.Rows(row).Cells(col).BottomBorderWidth = 2
                table.Rows(row).Cells(col).TopBorderWidth = 2
                table.Rows(row).Cells(col).LeftBorderWidth = 2
                table.Rows(row).Cells(col).RightBorderWidth = 2
                If col = cols - 1 Then
                        table.Rows(row).Cells(col).Alignment =
ContentAlignment.BottomRight
                End If
        Next
Next
```

C#

```
// 表を追加します
word.LineBreak();
int rows = 4;
int cols = 2;
RtfTable table = new RtfTable(rows, cols);
word.Add(table);
table.Rows[0].Cells[0].SetMerged(1, 2);

for (int row = 0; row < rows; row++) {
  if (row == 0) {
    table.Rows[row].Height = 50;
  }
  for (int col = 0; col < cols; col++) {
    if (row == 0 && col == 0) {
```

```
      text = Strings.DocumentBasicText2;
      table.Rows[row].Cells[col].Alignment =
ContentAlignment.MiddleCenter;
      table.Rows[row].Cells[col].BackFilling = Colors.LightPink;
    } else {
      text = string.Format("表のセル{0}:{1}", row, col);
      table.Rows[row].Cells[col].BackFilling = Colors.LightYellow;
    }
    table.Rows[row].Cells[col].Content.Add(new RtfString(text));
    table.Rows[row].Cells[col].BottomBorderWidth = 2;
    table.Rows[row].Cells[col].TopBorderWidth = 2;
    table.Rows[row].Cells[col].LeftBorderWidth = 2;
    table.Rows[row].Cells[col].RightBorderWidth = 2;
    if (col == cols - 1) {
      table.Rows[row].Cells[col].Alignment =
ContentAlignment.BottomRight;
    }
  }
}
```

上記のコードが作成する表は、Word ドキュメント内に適切にインデントされて配置されます。

上記のコードの出力は、次の図のようになります。

表のセル 1:0	表のセル 1:1
表のセル 2:0	表のセル 2:1
表のセル 3:0	表のセル 3:1

TOC の追加

Word for UWP では、テキストと見出しを含む Word ドキュメントに目次（TOC）を追加できます。これらの見出しを使用して TOC を作成します。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\UWP\WordSample
これらのクラスを上記の場所からアプリケーションで使用できます。

次のコードは、テキストと見出しを含む Word ドキュメントの TOC を作成します。

Visual Basic

```
Dim _doc As New C1WordDocument()

' タイトルを追加します
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
Dim rcPage As Rect = WordUtils.PageRectangle(doc)
Dim rc As Rect = WordUtils.RenderParagraph(doc, doc.Info.Title,
titleFont, rcPage, rcPage, False)
```

```
rc.Y += 12

' 意味がないドキュメントを作成します
Dim bkmk = New List()
Dim headerFont As New Font("Arial", 14, RtfFontStyle.Bold)
Dim bodyFont As New Font("Times New Roman", 11)
For i As Integer = 0 To 29
        ' i 番目の見出しを作成します(リンクターゲットおよびアウトラインエントリとして)
        Dim header As String = String.Format("{0}. {1}", i + 1,
BuildRandomTitle())
        rc = WordUtils.RenderParagraph(doc, header, headerFont, rcPage,
rc, True, _
                True)

        ' 後で TOC を構築するためにブックマークを保存します
        Dim pageNumber As Integer = doc.CurrentPage() + 1
        bkmk.Add(New String() {pageNumber.ToString(), header})

        ' テキストを作成します
        rc.X += 36
        rc.Width -= 36
        For j As Integer = 0 To 3 + (_rnd.[Next](20) - 1)
                Dim text As String = BuildRandomParagraph()
                rc = WordUtils.RenderParagraph(doc, text, bodyFont,
rcPage, rc)
                rc.Y += 6
        Next
        rc.X -= 36
        rc.Width += 36
        rc.Y += 20
Next

' TOC を開始します
doc.PageBreak()
' 新しいページで TOC を開始します
Dim tocPage As Integer = doc.CurrentPage()
' ページ索引を保存します(後で TOC から移動するため)
rc = WordUtils.RenderParagraph(doc,
Strings.TableOfContentsDocumentTitle, titleFont, rcPage, rcPage, True)
rc.Y += 12
rc.X += 30
rc.Width -= 40

' TOC をレンダリングします
Dim dottedPen As New Pen(Colors.Gray, 1.5F)
dottedPen.DashStyle = DashStyle.Dot
Dim sfRight As New StringFormat()
sfRight.Alignment = HorizontalAlignment.Right
rc.Height = bodyFont.Size * 1.2
For Each entry As String() In bkmk
        ' ブックマーク情報を取得します
        Dim page As String = entry(0)
```

```
        Dim header As String = entry(1)

        ' 見出し名とページ番号をレンダリングします
        doc.DrawString(header, bodyFont, Colors.Black, rc)
        doc.DrawString(page, bodyFont, Colors.Black, rc, sfRight)

        ' エントリにローカルハイパーリンクを追加します
        doc.AddLink(Strings.PoundSign + header)

        ' 次のエントリに移動します
        rc = WordUtils.Offset(rc, 0, rc.Height)
        If rc.Bottom > rcPage.Bottom Then
                doc.PageBreak()
                rc.Y = rcPage.Y
        End If
Next
```

C#

```
C1WordDocument _doc = new C1WordDocument();

// タイトルを追加します
Font titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
Rect rcPage = WordUtils.PageRectangle(doc);
Rect rc = WordUtils.RenderParagraph(doc, doc.Info.Title, titleFont,
rcPage, rcPage, false);
rc.Y += 12;

// 意味がないドキュメントを作成します
var bkmk = new List();
Font headerFont = new Font("Arial", 14, RtfFontStyle.Bold);
Font bodyFont = new Font("Times New Roman", 11);
for (int i = 0; i < 30; i++) {
  // i 番目の見出しを作成します(リンクターゲットおよびアウトラインエントリとして)
  string header = string.Format("{0}. {1}", i + 1, BuildRandomTitle());
  rc = WordUtils.RenderParagraph(doc, header, headerFont, rcPage, rc,
true, true);

  // 後で TOC を構築するためにブックマークを保存します
  int pageNumber = doc.CurrentPage() + 1;
  bkmk.Add(new string[] {
    pageNumber.ToString(), header
  });

  // テキストを作成します
  rc.X += 36;
  rc.Width -= 36;
  for (int j = 0; j < 3 + _rnd.Next(20); j++) {
    string text = BuildRandomParagraph();
    rc = WordUtils.RenderParagraph(doc, text, bodyFont, rcPage, rc);
    rc.Y += 6;
  }
  rc.X -= 36;
```

```
  rc.Width += 36;
  rc.Y += 20;
}

// TOC を開始します
doc.PageBreak();
// 新しいページで TOC を開始します
int tocPage = doc.CurrentPage();
// ページ索引を保存します(後で TOC から移動するため)
rc = WordUtils.RenderParagraph(doc,
Strings.TableOfContentsDocumentTitle, titleFont, rcPage, rcPage, true);
rc.Y += 12;
rc.X += 30;
rc.Width -= 40;

// TOC をレンダリングします
Pen dottedPen = new Pen(Colors.Gray, 1.5 f);
dottedPen.DashStyle = DashStyle.Dot;
StringFormat sfRight = new StringFormat();
sfRight.Alignment = HorizontalAlignment.Right;
rc.Height = bodyFont.Size * 1.2;
foreach(string[] entry in bkmk) {
  // ブックマーク情報を取得します
  string page = entry[0];
  string header = entry[1];

  // 見出し名とページ番号をレンダリングします
  doc.DrawString(header, bodyFont, Colors.Black, rc);
  doc.DrawString(page, bodyFont, Colors.Black, rc, sfRight);

  // エントリにローカルハイパーリンクを追加します
  doc.AddLink(Strings.PoundSign + header);

  // 次のエントリに移動します
  rc = WordUtils.Offset(rc, 0, rc.Height);
  if (rc.Bottom > rcPage.Bottom) {
    doc.PageBreak();
    rc.Y = rcPage.Y;
  }
}
```

上記のコードは、ドキュメント内のテキストの見出しにブックマークを追加します。次に、これらのブックマークを使用して TOC を生成します。

上記のコードの出力は、次の図のようになります。


UWP のコントロール

さまざまな用紙サイズの Word ドキュメントを作成

Word for UWP では、さまざまな用紙サイズで Word ドキュメントを作成できます。**PaperKind** 列挙を使用して、利用可能な標準用紙サイズを指定できます。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\UWP\WordSample
これらのクラスを上記の場所からアプリケーションで使用できます。

PaperKind 列挙は、次のコードで実装されます。

Visual Basic

```
' 各用紙サイズにつき 1 つのページを作成します
Dim firstPage As Boolean = True
For Each pk As PaperKind In [Enum].GetValues(GetType(PaperKind))
        ' Silverlight には Enum.GetValues はありません
        'PaperKind pk = fi;

        ' カスタムサイズはスキップします
        If pk = PaperKind.[Custom] Then
```

```
                Continue For
        End If

        ' 最初のページ以降のすべてのページに新しいページを追加します
        If Not firstPage Then
                word.PageBreak()
        End If
        firstPage = False

        ' 用紙の種類と向きを設定します
        'word.PaperKind = pk;
        word.Landscape = Not word.Landscape

        ' コンテンツをページに描画します
        rc = WordUtils.PageRectangle(word)
        rc = WordUtils.Inflate(rc, -6, -6)
        'string text = string.Format(Strings.StringFormatTwoArg,
word.PaperKind, word.Landscape);
        Dim text As String = String.Format(Strings.StringFormatTwoArg,
pk, word.Landscape)
        word.DrawString(text, font, Colors.Black, rc, sf)
        word.DrawRectangle(Colors.Black, rc)
Next
```

C#

```
// 各用紙サイズにつき 1 つのページを作成します
bool firstPage = true;
foreach(PaperKind pk in Enum.GetValues(typeof(PaperKind))) {
  // Silverlight には Enum.GetValues はありません
  //PaperKind pk = fi;

  // カスタムサイズはスキップします
  if (pk == PaperKind.Custom) continue;

  // 最初のページ以降のすべてのページに新しいページを追加します
  if (!firstPage) word.PageBreak();
  firstPage = false;

  // 用紙の種類と向きを設定します
  //word.PaperKind = pk;
  word.Landscape = !word.Landscape;

  // コンテンツをページに描画します
  rc = WordUtils.PageRectangle(word);
  rc = WordUtils.Inflate(rc, -6, -6);
  //string text = string.Format(Strings.StringFormatTwoArg,
word.PaperKind, word.Landscape);
  string text = string.Format(Strings.StringFormatTwoArg, pk,
word.Landscape);
  word.DrawString(text, font, Colors.Black, rc, sf);
  word.DrawRectangle(Colors.Black, rc);
}
```

テキストフローの追加

Word ドキュメントでテキストフローを使用できます。**Word for UWP** を使用して、ドキュメントの段やページにテキストをフロー挿入できます。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\UWP\WordSample
これらのクラスを上記の場所からアプリケーションで使用できます。

次のコードは、**Word for UWP** でテキストフロー機能を使用する方法を示します。

Visual Basic

```
 ' 長い文字列をリソースファイルからロードします
Dim text As String = Strings.ResourceNotFound

Using sr = New StreamReader(GetType(BasicTextPage).GetTypeInfo()

.Assembly.GetManifestResourceStream("WordSamples.Resources.flow.txt"))
        text = sr.ReadToEnd()
End Using
text = text.Replace(vbTab, "    ")

' Word ドキュメントを作成します
word.Info.Title = "テキストフロー"

' 表
Dim rows As Integer = 4
Dim cols As Integer = 2
Dim table As New RtfTable(rows, cols)
word.Add(table)
table.Rows(0).Cells(0).SetMerged(1, 2)

For row As Integer = 0 To rows - 1
        If row = 0 Then
                table.Rows(row).Height = 50
        End If
        For col As Integer = 0 To cols - 1
                'RtfParagraph paragraph = new RtfParagraph();
                'paragraph.Alignment = RtfHorizontalAlignment.Undefined;
                'paragraph.Content.Add(new
RtfString(string.Format("table cell {0}:{1}.", row, col)));
                'table.Rows[row].Cells[col].Content.Add(paragraph);
                table.Rows(row).Cells(col).Content.Add(New
RtfString(String.Format("table cell {0}:{1}.", row, col)))
                If row = 0 AndAlso col = 0 Then
                        table.Rows(row).Cells(col).Alignment =
ContentAlignment.MiddleCenter
                        table.Rows(row).Cells(col).BackFilling =
Colors.LightPink
                Else
                        table.Rows(row).Cells(col).BackFilling =
```

```
Colors.LightYellow
                End If
                table.Rows(row).Cells(col).BottomBorderWidth = 2
                table.Rows(row).Cells(col).TopBorderWidth = 2
                table.Rows(row).Cells(col).LeftBorderWidth = 2
                table.Rows(row).Cells(col).RightBorderWidth = 2
        Next
Next

Return

' タイトルを追加します
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
Dim bodyFont As New Font("Tahoma", 9)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim rc As Rect = WordUtils.RenderParagraph(word, word.Info.Title,
titleFont, rcPage, rcPage, False)
rc.Y += titleFont.Size + 6
rc.Height = rcPage.Height - rc.Y

' テキストに対して段を 2 つ作成します
Dim rcLeft As Rect = rc
rcLeft.Width = rcPage.Width / 2 - 12
rcLeft.Height = 300
rcLeft.Y = (rcPage.Y + rcPage.Height - rcLeft.Height) / 2
Dim rcRight As Rect = rcLeft
rcRight.X = rcPage.Right - rcRight.Width

' 左の段から開始します
rc = rcLeft

' 文字列を複数の段とページにまたがってレンダリングします
While True
        ' 四角形に収まるだけの文字列をレンダリングします
        rc = WordUtils.Inflate(rc, -3, -3)
        'int nextChar = word.DrawString(text, bodyFont, Colors.Black,
rc);
        word.DrawString(text, bodyFont, Colors.Black, rc)
        rc = WordUtils.Inflate(rc, +3, +3)
        word.DrawRectangle(Colors.LightGray, rc)

        ' 完了したら終了します
        'if (nextChar >= text.Length)
        If True Then
                Exit While
        End If

        ' レンダリングされた部分を削除します
        'text = text.Substring(nextChar);

        ' 右側の四角形に切り替えます
        If rc.Left = rcLeft.Left Then
```

```
                rc = rcRight
        Else
                ' 次のページの左側の四角形に切り替えます
                word.PageBreak()
                rc = rcLeft
        End If
End While
```

C#

```
// 長い文字列をリソースファイルからロードします
string text = Strings.ResourceNotFound;

using(var sr = new StreamReader(typeof(BasicTextPage).GetTypeInfo()

.Assembly.GetManifestResourceStream("WordSamples.Resources.flow.txt")))
{
  text = sr.ReadToEnd();
}
text = text.Replace("\t", "    ");

// Word ドキュメントを作成します
word.Info.Title = "テキストフロー";

// 表
int rows = 4;
int cols = 2;
RtfTable table = new RtfTable(rows, cols);
word.Add(table);
table.Rows[0].Cells[0].SetMerged(1, 2);

for (int row = 0; row < rows; row++) {
  if (row == 0) {
    table.Rows[row].Height = 50;
  }
  for (int col = 0; col < cols; col++) {
    //RtfParagraph paragraph = new RtfParagraph();
    //paragraph.Alignment = RtfHorizontalAlignment.Undefined;
    //paragraph.Content.Add(new RtfString(string.Format("table cell {0}:
{1}.", row, col)));
    //table.Rows[row].Cells[col].Content.Add(paragraph);
    table.Rows[row].Cells[col].Content.Add(new
RtfString(string.Format("table cell {0}:{1}.", row, col)));
    if (row == 0 && col == 0) {
      table.Rows[row].Cells[col].Alignment =
ContentAlignment.MiddleCenter;
      table.Rows[row].Cells[col].BackFilling = Colors.LightPink;
    } else {
      table.Rows[row].Cells[col].BackFilling = Colors.LightYellow;
    }
    table.Rows[row].Cells[col].BottomBorderWidth = 2;
    table.Rows[row].Cells[col].TopBorderWidth = 2;
    table.Rows[row].Cells[col].LeftBorderWidth = 2;
```

```
    table.Rows[row].Cells[col].RightBorderWidth = 2;
  }
}

return;

// タイトルを追加します
Font titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
Font bodyFont = new Font("Tahoma", 9);
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont,
rcPage, rcPage, false);
rc.Y += titleFont.Size + 6;
rc.Height = rcPage.Height - rc.Y;

// テキストに対して段を 2 つ作成します
Rect rcLeft = rc;
rcLeft.Width = rcPage.Width / 2 - 12;
rcLeft.Height = 300;
rcLeft.Y = (rcPage.Y + rcPage.Height - rcLeft.Height) / 2;
Rect rcRight = rcLeft;
rcRight.X = rcPage.Right - rcRight.Width;

// 左の段から開始します
rc = rcLeft;

// 文字列を複数の段とページにまたがってレンダリングします
for (;;) {
  // 四角形に収まるだけの文字列をレンダリングします
  rc = WordUtils.Inflate(rc, -3, -3);
  //int nextChar = word.DrawString(text, bodyFont, Colors.Black, rc);
  word.DrawString(text, bodyFont, Colors.Black, rc);
  rc = WordUtils.Inflate(rc, +3, +3);
  word.DrawRectangle(Colors.LightGray, rc);

  // 完了したら終了します
  //if (nextChar >= text.Length)
  {
    break;
  }

  // レンダリングされた部分を削除します
  //text = text.Substring(nextChar);

  // 右側の四角形に切り替えます
  if (rc.Left == rcLeft.Left) {
    rc = rcRight;
  } else // 次のページの左側の四角形に切り替えます
  {
    word.PageBreak();
    rc = rcLeft;
  }
```

```
}
```

上記のコードの出力は、次の図のようになります。

UI のエクスポート

Word for UWP を使用すると、ドキュメントに UI をエクスポートできます。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\UWP\WordSample
これらのクラスを上記の場所からアプリケーションで使用できます。

Word for UWP を使用して、UI を画像としてエクスポートし、次にその画像を Word ドキュメントで描画できます。UI を Word ドキュメントにエクスポートするには、次のコードを使用します。

Visual Basic

```
Dim _doc As New C1WordDocument()
_doc.Clear()
progressRing.IsActive = True
_doc.Landscape = True
panel.Arrange(_doc.PageRectangle())

' パネル内のすべての UI 要素を Word ドキュメントで描画します。
Await _doc.DrawElement(panel, _doc.PageRectangle())
WordUtils.SetDocumentInfo(_doc, Strings.RenderUIDocumentTitle)
WordUtils.Save(_doc)
```

```
progressRing.IsActive = False
```

C#

```
C1WordDocument _doc = new C1WordDocument();
_doc.Clear();
progressRing.IsActive = true;
_doc.Landscape = true;
panel.Arrange(_doc.PageRectangle());

// パネル内のすべての UI 要素を Word ドキュメントで描画します。
await _doc.DrawElement(panel, _doc.PageRectangle());
WordUtils.SetDocumentInfo(_doc, Strings.RenderUIDocumentTitle);
WordUtils.Save(_doc);
progressRing.IsActive = false;
```

上記のコードの出力は、次の図のようになります。