Understanding Document Rotation

Getting the Most from Solid Framework

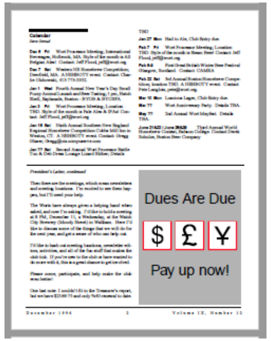
3rd July 2019

Solid Framework is an awesome tool for reconstructing documents from PDFs.

It can deal with both scanned and "Saved as" PDFs

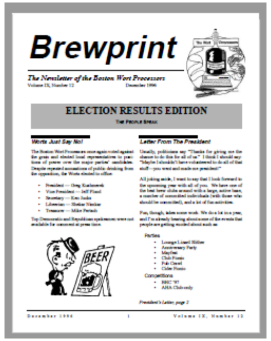
Solid Framework tries to help you

It is common when dealing with scanned documents to find that:
- some pages were scanned upside down, and others the right way up
- some pages were portrait and others were landscape format



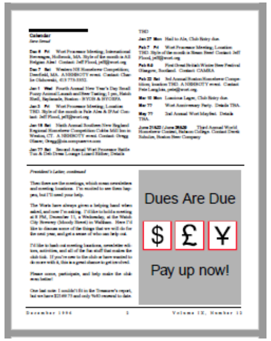
This can also happen with “Saved As” files that have been modified in a PDF editor

Solid Framework automatically detects the *dominant* text direction, and rotates the page so that text is horizontal


Brewprint

The Newsletter of the Boston Wort Processors

ELECTION RESULTS EDITION

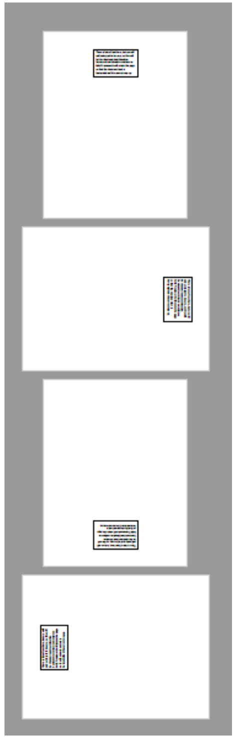

Dues Are Due

$ £ ¥

Pay up now!

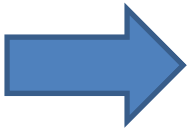
Rotation is on a page by page basis

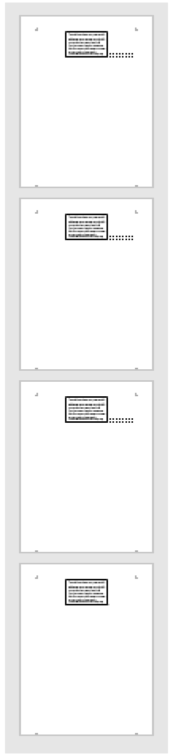
PDF



Same text rotated on each page

Word





All pages are the same way up

Sample file can be found at https://solidframework.net/wp-content/uploads/samples/textbox_orientation_multipage.pdf

This conversion is obvious when converting to Word, but the same effect occurs when a “Core Model” is created.

The Core Model is a structured “in-memory” representation of the PDF contents and offers many opportunities to deal with the *data* within the PDF without the need to convert to a Word Document.

Is this behaviour optional?

You can prevent the Converter from rotating a page left or right by 90 degrees (changing landscape to portrait or vice versa) using

```
converter.AutoRotate = false
```

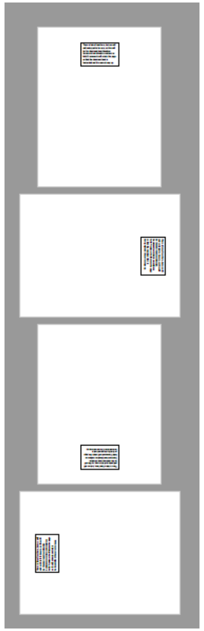
It is *not* possible to prevent pages being rotated by 180 degrees if the dominant text is upside down.

It is *not* possible to prevent this behaviour when using the Core Model.

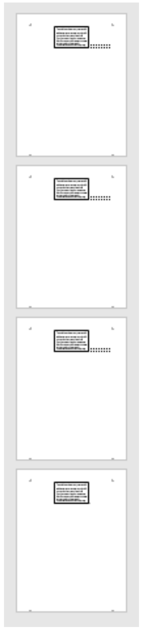
Identifying that page rotation occurred

GetPageWasAutoRotated(pageNumber) indicates the angle (in degrees clockwise) that the page was rotated in order to get it “the right way up”

PDF



Word



GetPageWasAutoRotated is 0

GetPageWasAutoRotated is 270

GetPageWasAutoRotated is 180

GetPageWasAutoRotated is 90

Note: “pageNumber” starts at one for the first page – even if a subset of pages were used to create the Core Model

What happens if there are different text orientations on the same page?

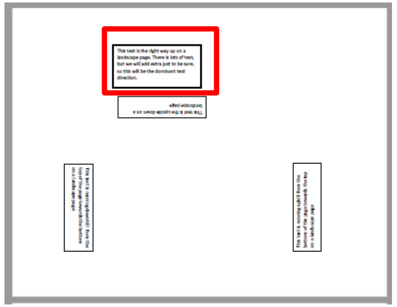
Solid Framework will detect the dominant text direction and rotate the page so that text is the right way up.

This can be a little confusing!

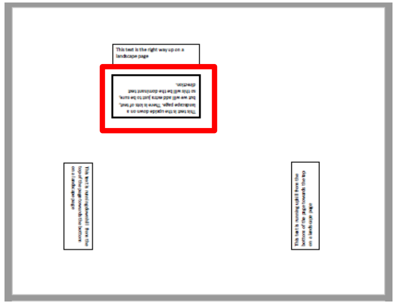
Dealing with different text orientations on the same page

The sample file contains fours pages

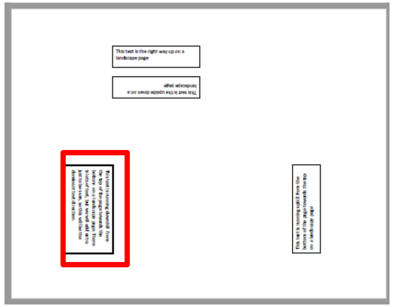
- each page has same general layout,
- one text box on each page contains more text which will make it dominate



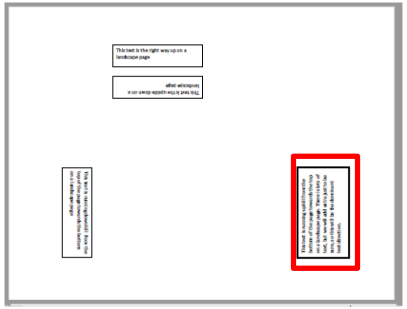
Dominant text is right way up



Dominant text is upside down



Dominant text is from top to bottom

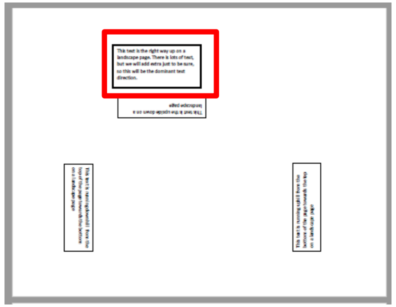


Dominant text is from bottom to top

Sample file can be found at https://solidframework.net/wp-content/uploads/samples/rotated_textboxes_landscape.pdf

Dominant text is right way up

PDF



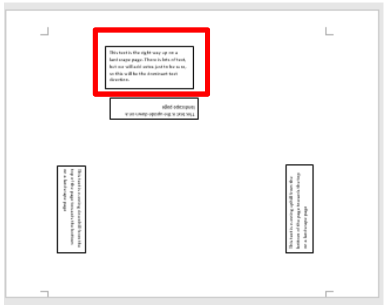
This text is the right way up on a landscape page. There is lots of text, but we will add extra just to be sure, so this will be the dominant text direction.

This text is the upside down on a landscape page

This text is running downhill from the top of the page towards the bottom on a landscape page

This text is running uphill from the bottom of the page towards the top on a landscape page

Word



This text is the right way up on a landscape page. There is lots of text, but we will add extra just to be sure, so this will be the dominant text direction.

This text is the upside down on a landscape page

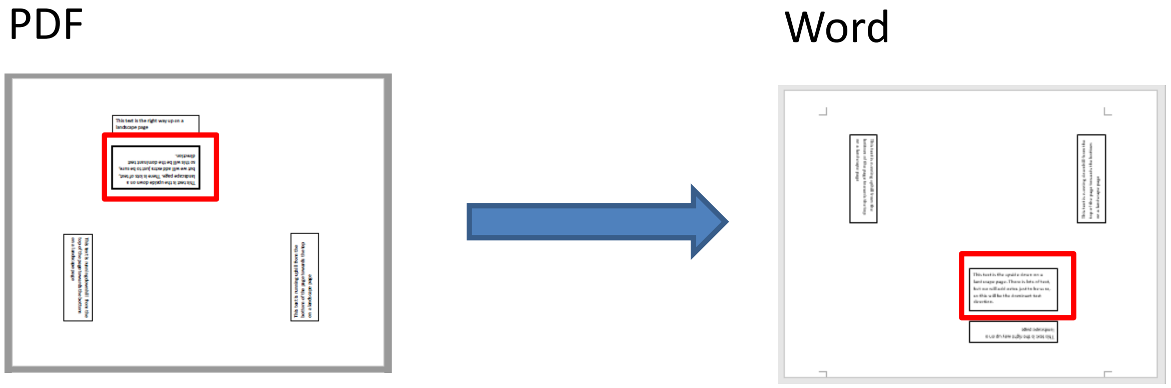
This text is running downhill from the top of the page towards the bottom on a landscape page

This text is running uphill from the bottom of the page towards the top on a landscape page

No change

GetPageWasAutoRotated = 0

Dominant text is upside down

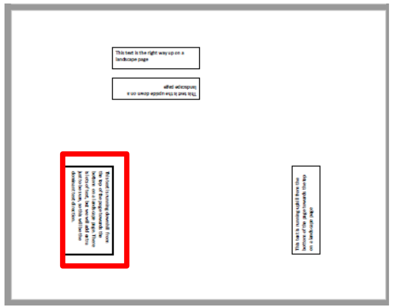


Page rotated upside down

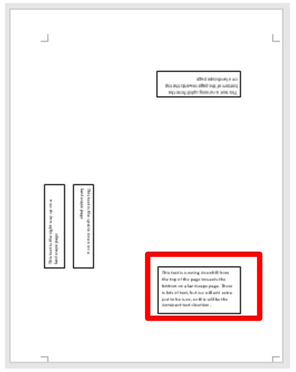
GetPageWasAutoRotated = 180

Dominant text runs from top to bottom

PDF


This text is the right way up on a landscape page
This text is the upside down on a landscape page
This text is running downhill from the top of the page towards the bottom on a landscape page. There is lots of text, but we will add extra just to be sure, so this will be the dominant text direction.
This text is running uphill from the bottom of the page towards the top on a landscape page

Word

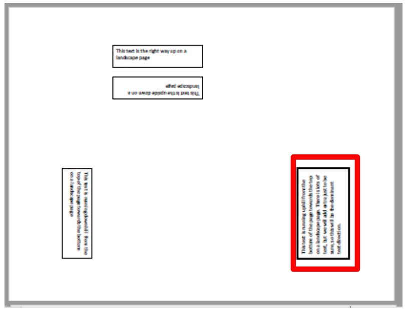


Page rotated left

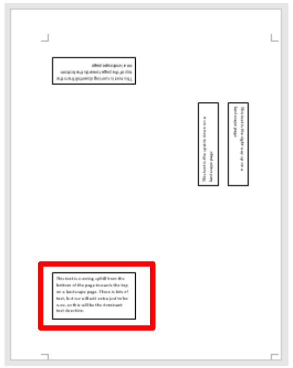
GetPageWasAutoRotated = 270

Dominant text runs from bottom to top

PDF


This text is the right way up on a landscape page
This text is the upside down on a landscape page
This text is running downhill from the top of the page towards the bottom on a landscape page
This text is running uphill from the bottom of the page towards the top on a landscape page. There is lots of text, but we will add extra just to be sure, so this will be the dominant text direction.

Word



Page rotated right

GetPageWasAutoRotated = 90

Identifying the orientation of text in the original PDF

Only applies to text within Text boxes

Logic for calculating the direction of text in the original document is complex.

Need to use:

Property	Provides information about
textbox.Rotation	If a text box has been rotated by 180 degrees
textbox.TextDirection	If the text is horizontal or vertical
GetPageWasAutoRotated(pageNumber)	If the page was rotated so that the dominant text is horizontal

Sample Code

```
private TextBoxRotation GetRotationForTextBox(SolidFramework.Model.Plumbing.TextBox tb, LayoutDocument ld, CoreModel cm)
{
    int workingRotation = 0;


    //the TextBox rotation is only really useful to identify whether the textbox has been rotated 180 degrees, since vertical text does not require the text box to be rotated.
    if (tb.Rotation > 0)
    {
        //Ensure the rotation is right-angled (divisible by 90), since other values are possible. This is most likely if the document was scanned.
        workingRotation = (int)(Math.Round((tb.Rotation / (double)90)) * 90) % 360;
    }

    //TextDirection.Horizontal may relate to text that was originally the right way up, or which was upside down.
    if (tb.TextDirection != TextDirection.Horizontal)
    {
        switch (tb.TextDirection)
        {
            case TextDirection.Rotate90:
                workingRotation = 90;
                break;
            case TextDirection.Rotate270:
                workingRotation = 270;
                break;
            default:
                break;
        }
    }

    int pageNumber = -1;
    int pageDir = -1;
    int initialRotation = -1;
    int finalRotation = -1;

    //Get the LayoutTextBox that contains layout information for the textbox. This is done as one method of getting the pagenumber
    LayoutTextBox ltb = ld.FindLayoutObject(tb.GetID()) as LayoutTextBox;
    if (ltb != null)
    {
        pageNumber = ltb.GetPageNumber();
        pageDir = cm.GetPageWasAutoRotated(pageNumber);

        initialRotation = (360 + workingRotation - pageDir) % 360;
        finalRotation =  workingRotation;
    }

    return new TextBoxRotation(tb.TextDirection, tb.Rotation, initialRotation, finalRotation, pageDir, pageNumber);
}
```

Summary

Solid Framework aims to minimise the number of choices that users have to make by setting logical defaults.

For most users this works well.

If required, the orientation of text can be calculated.