應用於手持裝置上之高效能電子書切頁加速定位演算法
An e-book Accelerate Positioning Paging Algorithm Used In Handheld Devices

蔡彰得 [1,a] 劉又齊 [1,b] 王卓彥 [1,c]
[1] 財團法人資訊工業策進會 創新應用服務研究所
Email: [a]ader@iii.org.tw, [b]allenycliu@iii.org.tw, [c]cylin@iii.org.tw

摘 要

近年來因國際電子圖書標準(ePub)順利地推行，成功帶動了數位閱讀市場有著顯著地成長，也改變了人們由傳統紙本轉為電子閱讀的習慣。在閱讀電子書的體驗上，經許多研究指出傳統紙本書的翻頁方式較捲軸滑動方式較為使用者接受，因而多數的電子書閱讀器均趨向以切頁式瀏覽模式為主。隨著數位內容的日益複雜，加上手持裝置因電子與成本考量故效能較低，使用者往往因需要等待閱讀器切頁的時間而犧牲了如傳統紙本翻頁的閱讀體驗。本研究提出一個支援 ePub 標準且可應用於手持裝置上之高效能電子書切頁加速定位演算法，方法中使用文件物件模型(Document Object Model, DOM)解析電子書內容，將每個檔案裡的顯示元件存成物件，每個物件存有其長、寬、物件型態、排版規則等資訊。將這些元件依序組合起來，先依裝置的顯示面積計算每行寬度予以換行，並以自行設計之排版引擎做排版。接著再依裝置顯示面積計算該頁的行數，並予以切頁，每切滿一頁即以回呼(Callback)的方式呼叫顯示引擎做該頁的內容呈現。另外，在切頁與排版的過程中本研究使用加速定位演算架構記錄每個元件之定位資訊，此架構可讓切頁引擎即時從該定位處做切頁運算而不需從頭算起，並在翻頁、跳頁、字體放大縮小等需要加速定位資訊時以此架構達到加速定位之效果。在實驗結果的部份，證實本論文所提出的演算法能夠有效達成切頁的目的，同時在提出的加速定位演算法之效能評估中，有使用加速定位法與未使用加速定位法在效能表現上有顯著差異。如在翻頁、跳頁、放大縮小字體等操作上能更有效率。未來本核心能擴增更多的顯示元件支援，使電子書的內容更具豐富性。

一、緣由與目的

數位時代的來臨，促成了網路的發展與普及，人們接受與發送訊息的方式更加多元，也進而改變了人們的生活品質。因為數位時代的來臨，造成人類越來越依賴科技產品：從 mp3 取代隨身聽與錄音機、DVD player 取代錄放影機、數位相機取代傳統相機、網路取代有線電視，到手機取代 mp3 隨身聽、相機，甚至桌上型電腦，資訊科技的發展速度之快可見一斑。

電腦作家 Steven Levey 曾預言，21 世紀將是實體印刷書籍的最後一個世紀[1];另一方面，Amazon 於 2007 年 11 月推出第一款 Kindle 電子書閱讀器，不到三年的時間，於 2010 年 7 月，亞馬遜宣布 Kindle 電子書的銷售量超過了精裝書籍。6 個月之後，Kindle 電子書的銷售量又超過了平裝書籍，從而成為亞馬遜網站上最受歡迎的書籍類型。2011 年 5 月 19 日，亞馬遜更宣佈 Kindle 電子書的銷售量超過精裝和平裝書籍的總和[2]。由此可知，我們傳統上所閱讀的實體書本將是這一波數位革命的主角。根據蒐集網路上電子書的翻頁模式，主要分為兩個使用者操作模式，一為捲軸式，二為翻頁式[3]。翻頁式相較於捲軸式比較受歡迎[4]，而在楊熾能的研究中發現，在閱讀速率方面，翻頁式明顯高於捲軸式閱讀[5]。由此可知，以翻頁方式閱讀電子書將會有較佳的使用者閱讀體驗。

二、相關研究

- ePub:為一個自由的開放標準，屬於一種可以「自動重新編排」的內容；也就是文字內容可以根據閱讀設備的特性，以最適於閱讀的方式顯示。EPub 檔案內部使用了 XHTML 或 DTBook （一種由 DAISY Consortium 提出的 XML 標準）來展現文字、並以 zip 壓縮格式來包裹檔案內容。EPub 格式中包含了數位版權管理（DRM）相關功能可供選用[6]。
- 文件物件模型(Document Object Model，簡稱 DOM):是 W3C 推薦的處理可延伸標示語言的標準程式介面。其概念是建立在將一份 HTML 或 XML 等相關文件視為一種 DOM，而該文件內的每個 tag 都是 DOM 中的一個元素(element)或節點(node)，例如:<html>即是 HTML 文件中的最高元素[7]。

三、系統實作

本研究採 Android 開發環境實作，開發工具為 Eclipse + ADT，以其套件提供之 Android 模擬器做測試，並以 ePub 格式的電子書為測試對象。

每一個 ePub 檔內有包含一至多個 Html 檔，並以 zip 格式壓縮。以 zip 解壓縮後會有 opf/ncx 資訊檔來記錄章節架構，並以每章節一個 html 檔呈現一到多個檔案。

本研究的核心部分在於以一個 html 檔為對象，對其進行解析、分頁，並加上定位資訊，最後測試有定位資訊與無定位資訊的效能差異。

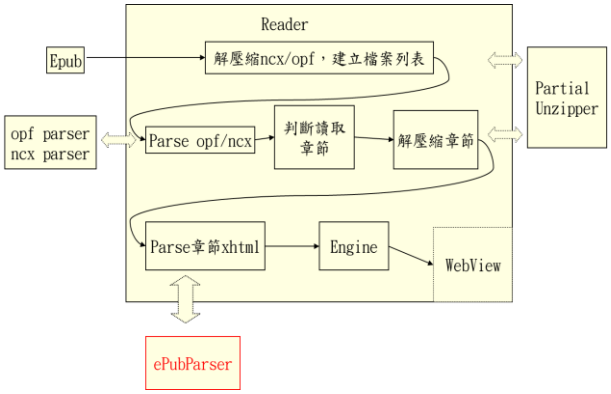

圖 1 ePub 處理流程圖

ePubParser 主要步驟如下:

1) 取得行動裝置螢幕之寬與高
2) 對 html 建立 DOM tree。建立完成之後，可對其每個 tag 逐一分析與運算。
3) 對有關顯示之標籤進行分析。針對 ePub 常見之顯示 tag,圖片與文字內容取得其寬與高的資訊，每取得一次就加入 ArrayList<DisplayComponent>物件中。再此以一持續增加之序號加入該物件之 index 變數中，做為未來定位使用。此處需注意的是 CSS 設定會影響物件的顯示大小，舉例來說文字的大小需以 CSS 設定，此處取得之文字高與寬也需將文字大小列入考慮來取得。

   DisplayComponent 物件為儲存相關資訊所用，其架構如下:

```
public class DisplayComponent {

    //定位索引
    private int index;
    //寬度
    private Double width;
    //高度
    private Double height;
    //html標籤
    private String tag;
    //是否換行
    private boolean isBreakLine;
    //是否換頁
    private boolean isBreakPage;

}
```

圖 2 DisplayComponent 物件內容

4) ArrayList<DisplayComponent>每增加一個單位就進行排版運算，看是否需要換行。文字的排版規則需與 Webview 一致。其排版運算也需將 CSS 等有關顯示之因素考慮在內，如字體放大縮小等，皆以 CSS 改變。
5) 當 ArrayList<DisplayComponent>行數達到切頁標準時，對主程式進行 Callback，將該頁的每個顯示元件以 html tag 格式組合成該頁內容，傳給 Webview 顯示。

```
public void epubParser(){

    int index=0;//定位資訊
    //取得螢幕長與寬
    //建立DOM tree
    while(!DOM節點結束){
        index++;
        if(node instanceof ImageTag){
            //取得圖片高與寬
        }else if(noce instanceof TextTag){
            //取得文字高與寬
        }
        //高度
        DisplayComponent.setHeight(height);
        //寬度
        DisplayComponent.setWidth(width);
        //定位資訊
        DisplayComponent.setIndex(index);
        //檢查排版設定
        if(checkBreakLine(ArrayList<DisplayComponent>)){
            //需換行
            DisplayComponent.setBreakLine(true);
        }
        if(checkCutPage(ArrayList<DisplayComponent>)){
            //檢查是否需切頁
            DisplayComponent.setBreakPage(true);
        }
        //增加至ArrayList中
        ArrayList<DisplayComponent>.add(DisplayComponent);

        if(DisplayComponent.isBreakPage==true){
            //分頁並callback至Webview
            makePage(ArrayList<DisplayComponent>);
        }
    }
}
```

圖 3 無加速定位之分頁流程虛擬碼

此方法的分頁程序是從一個 html 檔分成多個頁面，並從第一頁開始做分析。使用者在使用閱讀器時往往會有快速跳頁的功能需求，而行動裝置因為其硬體功能的限制，在跳頁時常常需耗費相當的時間才能到目的頁面，這時可使用定位資訊來加速定位。

每一節點都有保留其自己的定位資訊，當一本 ePub 書第一次分頁完成後，每一個分頁的第一個定位索引即為該頁的定位索引。如果以後需要跳至該頁，只要以該頁的定位索引為參數，並在分頁的一開始判斷是否以達到該索引，如果未到達則不需運算；已到達之後才進行正常的分頁程序，如此可省下大量的運算資源。

```
public void epubParser(int locateIndex){

    int index=0;//定位資訊
    //取得螢幕長與寬
    //建立DOM tree
    while(!DOM節點結束){
        index++;
        //當index大於定位資訊時，才進行分頁運算
        if(locateIndex>=index){
            if(node instanceof ImageTag){
                //取得圖片高與寬
            }else if(noce instanceof TextTag){
                //取得文字高與寬
            }
            //高
            DisplayComponent.setHeight(height);
            //寬
            DisplayComponent.setWidth(width);
            //定位資訊
            DisplayComponent.setIndex(index);
            //檢查排版設定
            if(checkBreakLine(ArrayList<DisplayComponent>)){
                //需換行
                DisplayComponent.setBreakLine(true);
            }
            if(checkCutPage(ArrayList<DisplayComponent>)){
                //檢查是否需切頁
                DisplayComponent.setBreakPage(true);
            }
            //增加至ArrayList中
            ArrayList<DisplayComponent>.add(DisplayComponent);
            if(DisplayComponent.isBreakPage==true){
                //分頁並callback至Webview
                makePage(ArrayList<DisplayComponent>);
            }
        }
    }

}
```

圖 4 有加速定位之分頁流程虛擬碼

四、實驗結果

本研究提出了以定位索引來加速跳頁的速度，實驗將以同一本 ePub 書的某一章節來做測試。此實驗將以一本 ePub 電子書--小王子來做測試, 實驗 1：以 A 組與 B 組測試兩者跳至同一頁的速度差異；實驗 2：以 A 組與 B 組測試兩者於第 5 頁放大字體的速度差異；實驗 3：以 A 組與 B 組測試兩者於第 5 頁縮小字體的速度差異；實驗 4：以 A 組與 B 組測試兩者跳至上一章最後一頁的速度差異；以上實驗之測試單位均為毫秒(ms)。

表 1 實驗結果

	A 組-使用正常分頁	B 組-使用加速定位分頁
實驗 1:跳至第 10 頁	1376ms	234ms
實驗 2:於第 5 頁字體放大	757ms	213ms
實驗 3:於第 5 頁字體縮小	533ms	169ms
實驗 4:跳至上一章的最後一頁	1501ms	254ms

五、結論

本研究提出了一個分頁加速定位方法，可讓 ePub 電子書的分頁功能更趨近於使用實體書籍的使用者體驗。實驗結果發現，使用此一加速定位方法有顯著的加速效果，雖然只測試了圖片及文字顯示物件，但對於其他的顯示物件如影像、flash 應也可適用，此部分目前還在研究中，可望在未來加入其他的顯示物件支援。

六、誌謝

本研究依經濟部補助財團法人資訊工業策進會「100 年度數位匯流服務開放平台技術研發計畫(2/4)」辦理。

參考文獻

[1] 錢愛玲，「電子書--圖書市場的未來」，高等函授學報14卷4期(2001年8月):頁64

[2] STPI 科計產業資訊室，http://cdnet.stpi.org.tw/techroom/market/ee/2011/ee_11_007.htm,[May,13,2011]

[3] 張雅慈，「高齡者閱讀小型螢幕電子書之研究」，2010年，雲林科技大學工業設計學系，碩士論文。

[4] Millis, C.B., & Weldon, L.J., “Reading text from computer screens”, *ACM Computing Surveys*, vol.19(4), pp.329-357, 1987.

[5] 楊熾能，「小螢幕文本呈現方式與視窗尺寸對閱讀速率、理解率、閱讀效率、滿意度之效應研究」，2006年，天主教輔仁大學資訊管理學系，碩士論文。

[6] epubSpecifications IDPF, http://idpf.org/, [July 2, 2009].

[7] W3C Document Object Model , http://www.w3.org/DOM/