第二章、相關研究

本章將會介紹與本研究有關的技術以及相關文獻的探討。第一節會先介紹遊戲的開發流程。第二節接著會介紹由本實驗室所開發的 3D 遊戲編輯器。第三節則是會對手持式裝置上可以使用的 3D 繪圖函式庫作一個簡單的介紹以及分析。

2.1 遊戲開發分工與流程

在本節之中會先針對遊戲的開發流程，以及開發遊戲時人員的分工情況加以簡單的介紹。而關於遊戲的開發流程之研究有相當多的文獻資料可以參考，如〔12 ,13 ,14 ,15, 16, 17, 18, 19〕。

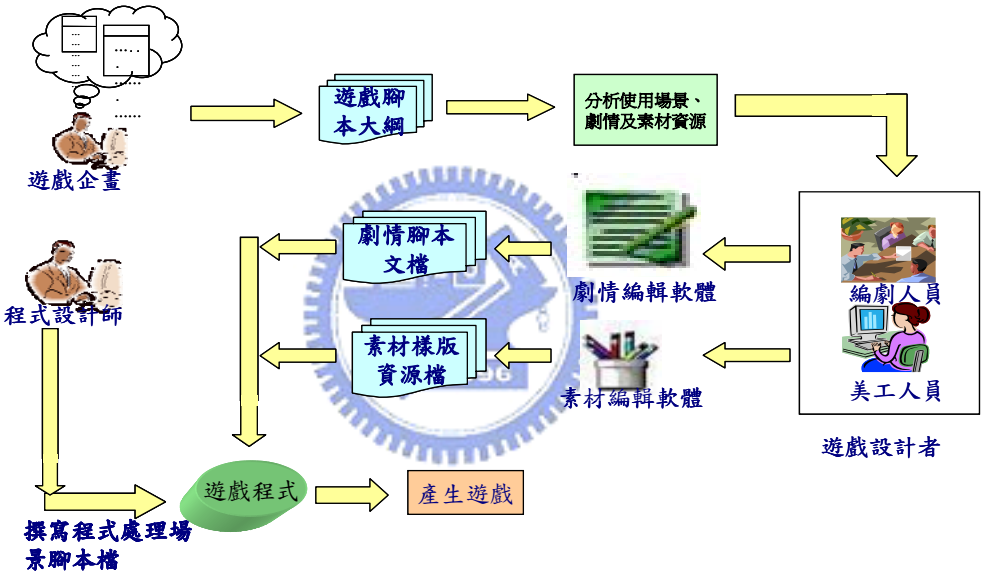

圖 1 傳統的遊戲開發流程〔10〕

圖 1 展示了一個傳統上的遊戲開發流程。遊戲的開發人員大致上可以分為三類〔11〕:

1. 遊戲企劃人員

   遊戲企劃人員是一個遊戲的創始者。遊戲企劃人員可根據目前遊戲市場上之遊戲需求類型調查，發揮自己的創意，提出所謂的遊戲腳本大綱，也是就是遊戲的主要架構。以便可以與遊戲設計人員討論遊戲需要的素材以及遊戲的可行度。

2. 遊戲設計人員

   遊戲設計人員則是負責將遊戲需要的材料準備好的角色。當遊戲企劃人員提出遊戲腳本大綱後，便與遊戲設計人員及遊戲程式設計人員討論如何將遊戲實現出來。

當討論結束之後，根據討論結果，遊戲中所需要的多媒體素材資源交由遊戲設計人員中的美工人才進行設定，包含角色的外型設定、遊戲畫面的設定。而遊戲的腳本則交由編劇人員負責，將遊戲企劃人員的遊戲腳本大綱，分割成一個一個遊戲場景的分鏡，然後將遊戲的場景通通寫成遊戲描述檔。這個描述檔不是像日常所使用的文字描述一般，而是用一些格式化的格式來進行整個遊戲場景的描述。遊戲場景描述檔中會針對遊戲場景中所使用到的遊戲素材，以及在會在該場景中出現的事件加以描述。如圖 2 所示：

```
[bgmusic]
mid_field.mid
[pic]
//bmp index,bmp name,width,height,attribute
0 forest2_ground.bmp 30 30 0
1 forest2_rock1.bmp 30 30 0
2 forest2_rock2.bmp 30 30 0
3 forest2_rock3.bmp 30 30 0
[map]
//map size
30 30
00000000000000000000000000000
0
00000000000000000000000000000
0
00000000000000000000000000000
0
00000000000000000000000000000
0
00000000000000000000000000000
0
00000000000000000000020000010
0
00000003000000000000000000000
0
00000000000000000000000000000
```

圖 2 以遊戲描述檔描述遊戲場景的範例〔10〕

當遊戲設計所需要的多媒體素材以及遊戲描述檔都到齊之後，接下來的工作就交給遊戲程式設計人員負責。

3. 程式設計人員

遊戲程式設計人員負責的就是將遊戲設計人員所設計出來的遊戲素材與遊戲組合起來，製作成一個能夠給玩家進行遊玩的遊戲。也是遊戲能否吸引人的重要角色之一，所有玩家在遊戲中能見到特效都仰賴程式設計人員的巧思設計。

在傳統的遊戲開發流程中存在一些問題：

1. 程式設計人員與遊戲設計人員之間溝通的問題

   在圖 2 中的例子中可以看到，遊戲設計人員在使用遊戲描述語言來描述遊戲場景時，會有不夠直覺的問題。這樣會造成劇情編輯人員在編輯時，必須時時注意到語法的格式是否錯誤。另外，遊戲描述語言的格式是由程式設計人員制定，如此也會造成劇情編輯人員在編輯上的不方便，若程式設計人員修改語法的話，劇情編輯人員就必須將已經編輯完成的描述

檔進行修改，這會造成許多開發上的不便。

2. 素材資源重複利用的問題

當遊戲開發結束之後，所設計的遊戲多媒體素材假如沒有一個好的機制作管理，可能在遊戲開發結束後就被棄置。不但造成浪費，而且在之後的遊戲開發上也無法有效的再利用，使得每次遊戲開發都必須再度開發一次類似的素材資源。

因此，本實驗室根據以上所提的缺點，經過卓勇君〔10〕、曾華堃〔11〕等幾位學長的努力，研發出以視覺化編輯取代使用撰寫遊戲描述的方式開發遊戲，並且提供素材的管理機制，讓美工人員的努力得以重複利用。在下一小節中將會對本實驗室所研發出的 3D 角色扮演遊戲編輯器作介紹。

2.2 3D 角色扮演遊戲編輯器介紹

首先，先來看看 3D 角色扮演遊戲編輯器所使用的視覺化遊戲開發流程。

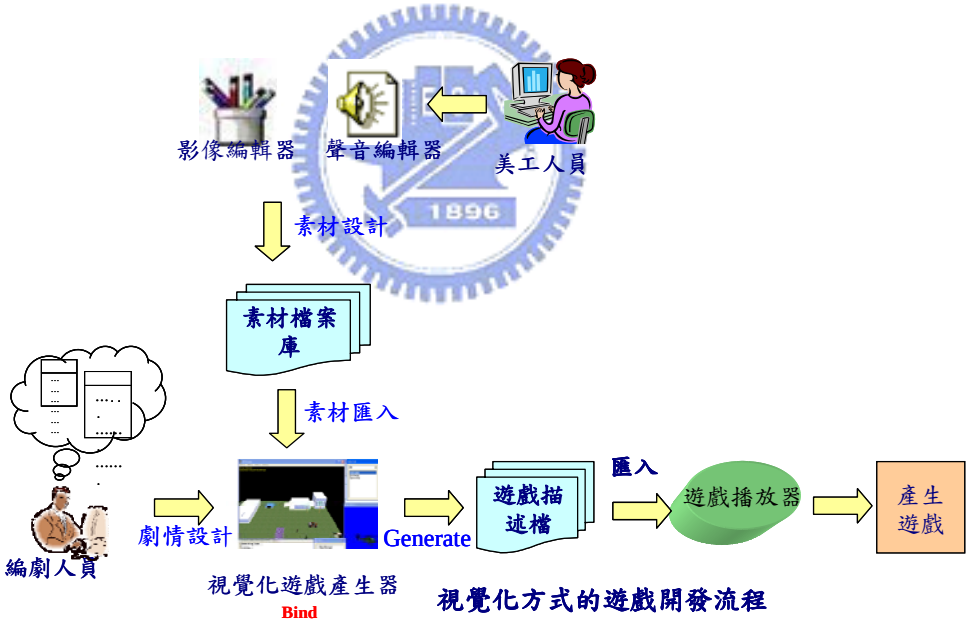

圖 3 視覺化遊戲開發流程〔10〕

可以將圖 1 與圖 3 做個比較。在視覺化的遊戲開發流程之下，可以發現缺少了程式設計人員的參加。這是因為程式設計人員在傳統的遊戲開發流程中的功用，被視覺化遊戲產生器給取代。劇情編輯人員利用視覺化遊戲編輯器就可以進行遊戲的編輯，不用耗費心力在遊戲描述檔的撰寫上。而程式設計人員的工作就由遊戲的程式設計轉成遊戲的編輯工具的開發。

同時可以發現，在圖 3 中增加了素材檔案的管理機制，遊戲美術人員利用各種編輯

器產生出來的心血結晶，通通可以存入素材檔案資源庫中，透過素材檔案資源庫管理這些遊戲素材，讓劇情編輯人員可以輕易的找到想要使用的遊戲素材，並使用在遊戲之中。而每當遊戲開發完畢，所設計出來的多媒體素材還可以提供給下一個新的遊戲開發來使用。

下圖則是使用 3D 遊戲編輯器來製作遊戲的一個範例：

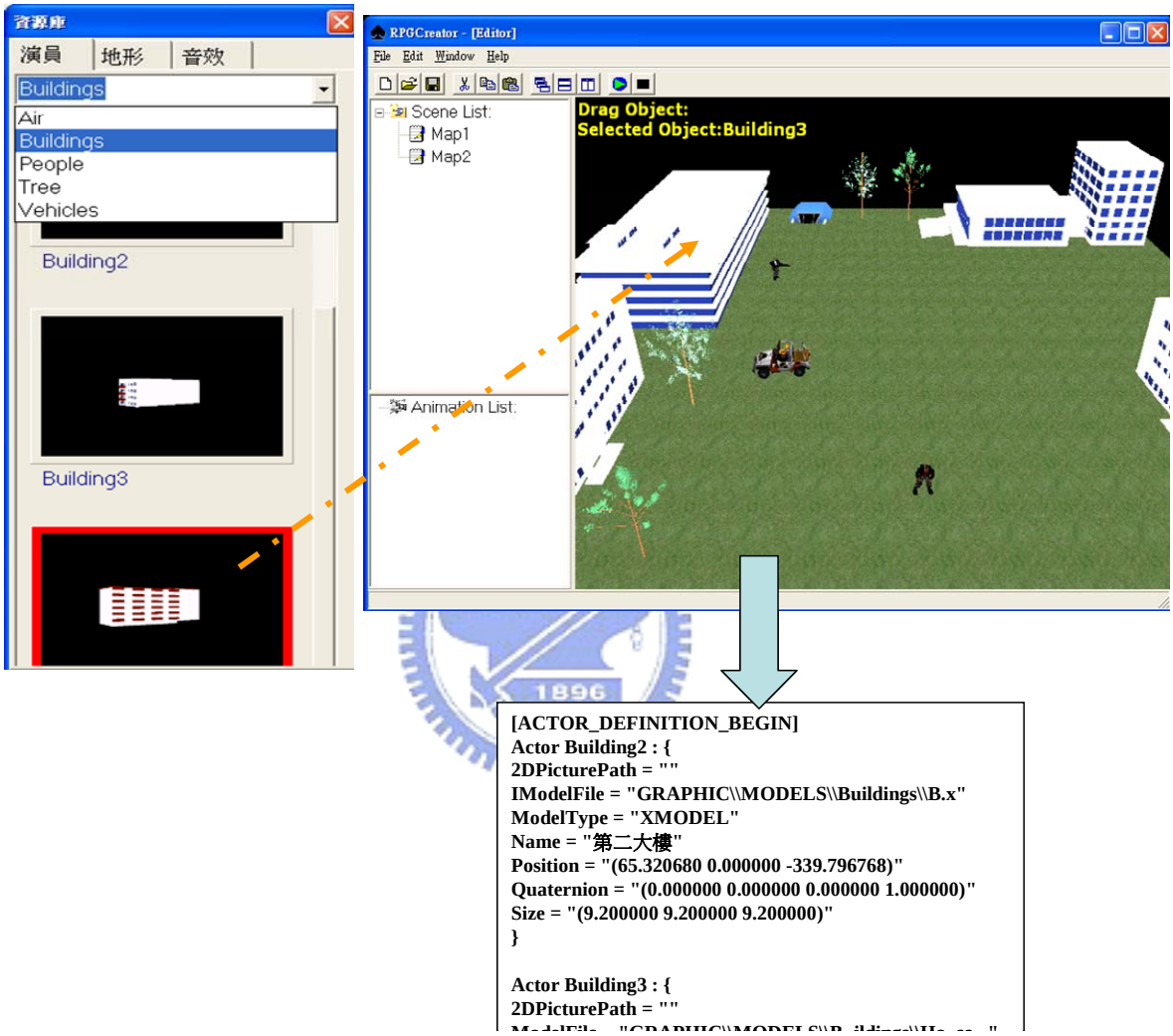

圖 4 以視覺化編輯器製作遊戲的範例〔10〕

圖 4 中左邊可以看到資源庫，資源庫依照「演員」、「地形」、「音效」三大類分別放置，在各大類中還依照不同的性質分成幾個小類，讓編輯人員可以迅速的找到他想用的素材資源。圖中右邊則是遊戲編輯器，編輯人員可以直接從素材庫中拖拉素材到編輯器中，依照遊戲腳本的需求去建立遊戲場景。

當編輯人員編輯完成整個遊戲的場景之後，遊戲編輯器會自動的轉換成遊戲描述檔的格式儲存。最後，將遊戲描述檔交給遊戲播放器來執行，便可以產生一個實際可以遊玩的遊戲了。

總而言之，3D 角色扮演遊戲編輯器可以歸納出以下幾個特點：

1. 簡化編輯遊戲劇情的設定方式

   透過視覺化編輯的「所見即所得」方式，省卻劇情編輯人員在編輯遊戲劇情的時候需要去記憶描述檔語法，免除了語法使用上的錯誤。另外，也減少了當描述語法修改時，需要對描述檔進行修改的困難度。

2. 減輕程式設計師的負擔

   有時候當程式設計人員將遊戲完成之後，遊戲企劃人員會認為與當初的設定有所差距，而會要求重新修改，最後又得交由程式設計人員在重新產生遊戲。這樣的流程不但浪費資源及時間，連帶著增加程式設計人員的工作。使用視覺化編輯器就可以有效減少這樣的情況發生。

因此，本實驗室所開發出來的 3D 遊戲編輯器的確有效的解決在傳統遊戲開發上的問題，連帶也可以提高在開發遊戲時的效益。

2.3 使用於手持裝置上的 3D 繪圖應用程式介面簡介

因為本研究的目的是希望利用視覺化的編輯環境產生可以在手持式裝置上的 3D 腳色扮演遊戲。所以，想要在手持式裝置上繪製 3D 物件的話，就勢必需要一個 3D 的繪圖應用程式介面（Application Program interface，簡稱 API）來幫助我們達到這目的。因此，在本小節中，將會對目前比較常見的且適合用在手持式裝置上的 3D 繪圖應用程式介面作一個簡單的介紹。

2.3.1 OpenGL ES 簡介〔24〕

OpenGL 是相當知名且歷史悠久的 2D/3D 繪圖 API。OpenGL 的優點在於其需求低、效能高、取得方便(OpenGL 幾乎是免費的)，而且為業界標準之一，目前在眾多 2D/3D 繪圖 API 中頗受到廠商的歡迎。

OpenGL ES 則是特別針對嵌入式系統，從 OpenGL 中萃取出來的繪圖用 API。其架構（Framework）如下所述：

1. 協定（Profile）

   OpenGL ES 以定義協定的方式來對應到各種不同的應用環境上。目前 OpenGL ES 的規格中分為兩類協定：

   a. 一般協定(Common Profile)

      為了娛樂消費平台而設定，具有最大的相容性。應用的範圍有 PDA、遊戲機等等。

   b. 安全協定(Safety Critical Profile)

      針對特殊應用產品而設定，強調產品的可靠度以及安全性。可應用在航空或是汽車的顯示技術。

2. 一致性（Conformance）測試

每年 OpenGL 都會檢討修訂規格，為了維持產品與 OpenGL 的新規格相容，就必須經過一致性的測試，而這些測試也會註明在 OpenGL ES 的規格文件中，以確保所有使用 OpenGL ES 所開發出來的產品也同樣可以符合新規格的規範。

3. 延伸指令集(Extensions)

OpenGL ES 可以用加入延伸指令集的方式提供新的功能，OpenGL ES 本身也提供了一份延伸指令集供廠商選用。而這份延伸指令集已經經過一制性的測試，以確保符合 OpenGL 所制定的規格。假若廠商選擇使用自己的延伸指令集的話，同樣也需要經過一制性測試後才可以在產品上標明與 OpenGL 相容。

4. 原生平台繪圖介面層(Native Platform Graphics Interface Layer)

原生平台繪圖介面層與平台間是互相獨立的。廠商可以決定是否在產品中加入此一介面幫助產品的開發，也可以選擇自行開發介面層來使用。

談完了 OpenGL ES 的架構，接著下面來探討使用 OpenGL ES 來開發嵌入式系統上的應用程式有哪些優點存在：

1. 產業標準及免權利金

OpenGL ES 既然是繼承了 OpenGL 而來，當然也繼承了 OpenGL 的性質－產業標準以及免權利金。這讓 OpenGL ES 擁有相當的吸引力，吸引應用程式開發者來使用 OpenGL ES 進行應用程式的開發。

2. 小容量及低耗電量

由於嵌入式系統的規格差異頗大，不同的硬體之間其運算能力以及儲存容量都不盡相同。OpenGL ES 對這一特點考量之後，在設計上就令 OpenGL ES 只需要非常少的儲存容量跟極低的運算需求，就可以讓它輕鬆的使用在嵌入式系統之上。

3. 延伸和未來發展

OpenGL ES 利用延伸指令的方式，讓開發者可以自由的對 Open GL 增加新的特色，如支援硬體所提供的特殊應用，或是增加新的功能。

4. 充分的文件支援

由於 OpenGL ES 是從 OpenGL 而生。因此，過去在 OpenGL 發展時期所留下的所有教學說明、書籍以及範例程式，是非常大量而且依然具有參考價值的。因此，學習如何使用 OpenGL ES 來進行開發變的非常方便。

從以上的優點看來，OpenGL ES 可以說是挾著 OpenGL 原來的優勢轉攻嵌入式系統。也因此，在目前的嵌入式系統中，許多廠商看上了 OpenGL ES 由 Open GL 繼承而來的優點，乃採用 OpenGL ES 作為產品上的繪圖 API，如新力（Sony）公司所開發出來的掌上型遊戲機 PSP，其繪圖 API 就是採用 OpenGL ES。

2.3.2 J2ME 的 JSR184 簡介〔25〕

在上一節提到的 OpenGL ES，是由所有製造繪圖晶片以及研發繪圖軟體的廠商共同討論出來的產物。在本節中，將要討論另外一個由手機製造廠商與昇陽（Sun）共同推出的新的 3D 繪圖 API－JSR184。

在 Sun 的 Java 規劃藍圖中，Jave ME（Java Platform, Micro Edition）是專門針對嵌入式系統所設計，尤其是行動裝置，如個人行動助理（PDA）、手機。而 JSR-184 則是在 J2ME 下專門提供 3D 繪圖使用的 API。其架構圖如下所示：

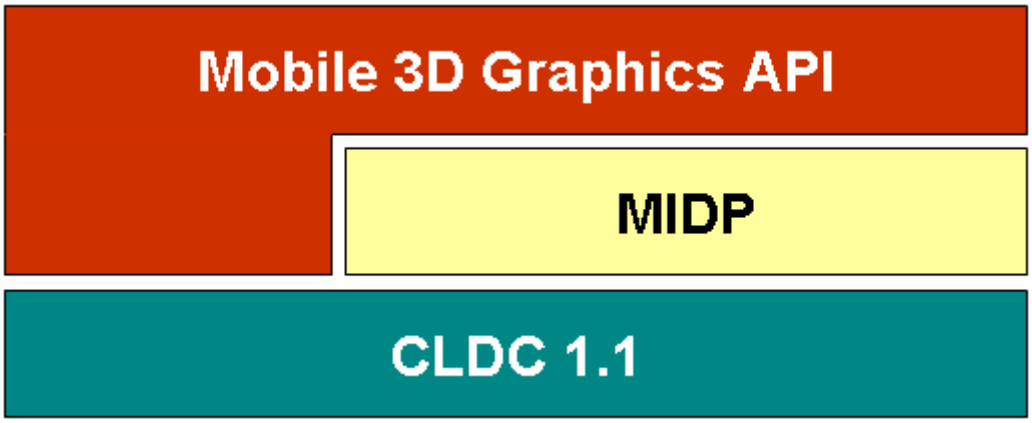

圖 5 JSR-184 架構圖

JSR-184 的特色如下：

1.具備兩種繪圖模式－保留模式（retained mode）以及立即模式（immediate mode），以便應用程式開發者挑選適合的繪圖方式來應用。

2.可與 OpenGL ES 相容。

3.所開發出來的程式，具有容量小的特性，方便在嵌入式系統中應用。

4.具有資源回收的機制（garbage-collection）。

在這邊特別要說明的就是 JSR-184 所提供的兩種繪圖模式，茲說明如下：

1.保留模式（retained mode，簡稱 RM）

在 RM 模式中，3D 繪圖的資料是使用 M3G 格式的檔案來提供。M3G 格式是在 JSR-184 規格中所提出的檔案格式，用於儲存 3D 場景的資料，並且利用二進位的編碼方式儲存。使用二進位來編碼的原因是因為這樣做可以減少整個 3D 資料的大小，並且提高保密性，讓裡面包含的 3D 資料不容易被外人所取得。但是，相對的想要編輯 M3G 格式的資料就必須使用特殊的編輯器才可以進行編輯修改。表格 1 是 M3G 檔案的架構。

表格 1　M3G 檔案架構〔25〕

	File Identifier
Section 0	File Header Object
Section 1	External Reference Objects
Section 2	Scene Objects
Section 3	Scene Objects
...	...
Section *n*	Scene Objects

M3G 的檔案架構基本上就是由一個檔案識別碼（File Identifier）後面跟著連續幾個節（Section），每節中則擺放了 3D 的繪圖資料。至於詳細的說明，在本論文中就不詳加討論。

由於 M3G 檔案是以場景為單位來儲存 3D 資料，因此將 M3G 格式的檔案使用在遊戲製作上，可以說是事半功倍。不過，相對的 M3G 的格式也成為目前使用上的最大阻礙。支援 M3G 格式的 3D 編輯器在目前還是少數，因此 M3G 在使用上還是不夠方便，這也使得使用 M3G 格式來製作遊戲上變的複雜。

以下是使用 RM 模式繪圖的範例：

```
1    try{
2          myWorld = (World)Loader.load
3          ("/com/superscape/m3g/wtksamples/retainedmode/content/swerve.m3g")〔0〕;
4          setupAspectRatio();
5    }catch(Exception e)
6    {e.printStackTrace();}
7    worldStartTime = System.currentTimeMillis();
8    myCanvas.repaint();
```

M3G 格式使用 http 的協定載入，這令 RM 模式下的繪圖資料來源，不只是從手機本身的記憶體中取得，也可以透過網路經由下載的方式取得繪圖資料。在第二行中，使用透過 ISR-184 所提供的函式－LoaderLoad，將事先編輯完成的 3D 場景載入到程式之中，然後播放在手機的螢幕之上。下圖是模擬器執行的畫面：

圖 6 RM 模式的範例執行畫面

使用 RM 模式繪圖，可以明顯的看到畫面上的呈現會比較細緻，因為這是事先透過其他編輯器所編輯出來的場景畫面。所以在前面才會說 RM 模式比較適合於遊戲的製作。

2.立即模式（immediate mode，簡稱 IM）

IM 模式的 3D 繪圖資料來源則與 RM 模式的不同，IM 模式所繪製的 3D 場景資料是直接置放於繪圖程式之中，與繪圖程式本身有著切不可分的關係。

IM 模式的優點在於，3D 的資料已經存在於程式之中，可以很明確的知道程式所使用到的記憶體大小以及程式可以針對資料作最佳化處理。相較於 RM 模式而言，3D 資料的來源不需要在從其他地方載入，可以節省許多載入場景的時間。但是相對的，使用在程式中的 3D 資料就必須注意到 J2ME 在程式的記憶體上所設定的顯制。目前 J2ME 設定在手機上可執行的程式最大只能有 64k，而在 3D 物件的多邊型的頂點數目須在 3000 點以下。超過這些限制的話，就無法使用。下面則是使用 IM 模式繪圖的範例：

```
1   //3D Data
2   int〔〕〔〕 aaStripLengths = {{4}, {4}, {4}, {4}, {4}, {4}};
3   short〔〕 aPos = { -1, -1, 1, 1, -1, 1, ..., -1, 1, 1};
4   byte〔〕 aCol = {     -1, 0, 0, -1, 0, 0, ..., -1, 0, 0};
5   int cSubmeshes = aaStripLengths.length, cVertices = aPos.length/3;
6   //Initial 3D Object
7   vertexBuffer.setDefaultColor(0xFFFFFFFF);
8   vaPos.set(0, cVertices, aPos);
9   vertexBuffer.setPositions(vaPos, 0.40f, null);
10  vaCols.set(0, cVertices, aCol);
11  vertexBuffer.setColors(vaCols);
```

```
12  for(int i=0;i<cSubmeshes;i++){
13          appa〔i〕=app;
14          iba〔i〕 = new TriangleStripArray(startIndex, aaStripLengths〔i〕);
15          for(int j=0; j<aaStripLengths〔i〕.length;j++)
16                  startIndex+=aaStripLengths〔i〕〔j〕;
17  }
18  //Paint to Canvas
19  myWorld = new World();
20  Group rootGroup2 = new Group();
21  myWorld.addChild(rootGroup2);
22  rootGroup2.setOrientation(15.0f,1.0f,0.0f,0.0f);
23  rootGroup = new Group();
24  rootGroup2.addChild(rootGroup);
25  //Set Camera
26  Camera myCamera = new Camera();
27  myWorld.addChild(myCamera);
28  myWorld.setActiveCamera(myCamera);
29  myCamera.setParallel(CUBESIZE*1.5f, 1.0f, -CUBESIZE, CUBESIZE);
30  int index = 0;
31  for(int i=0; i<CUBESIZE; i++){
32          float x = (i*2 - CUBESIZE) * 0.5f;
33          for(int j=0; j<CUBESIZE; j++){
34                  float y = (j*2 - CUBESIZE) * 0.5f;
35                  for(int k=0; k<CUBESIZE; k++){
36                          float z = (k*2 - CUBESIZE) * 0.5f;
37                          Mesh m = new Mesh(vertexBuffer, iba, appa);
38                          m.setTranslation(x,y,z);
39                          rootGroup.addChild(m);
40                          currentState〔index〕=(rand.nextInt()>0x40000000)?(byte)1:(byte)0;
41                          cells〔index++〕 = m;
42  }}}
43  myCanvas.repaint();
```

這段範例是亂數且隨時間而更動的在螢幕上繪製出一群立方體，前面 1～5 行就是接下來繪圖程式所要繪製的 3D 物件資料，也就是每個立方體上的頂點及包含邊之 3D 資料。6～17 行則是 3D 物件初始化的處裡，在這段程式碼中，設定立方體的顏色、材質、以及光影效果。而接下來的 18~43 行就是將亂數產生的立方體畫在

螢幕的畫布上，其中 31~42 行是負責計算正方體由亂數所產生的繪製位置。下圖是此範例在手機模擬器上執行的結果：

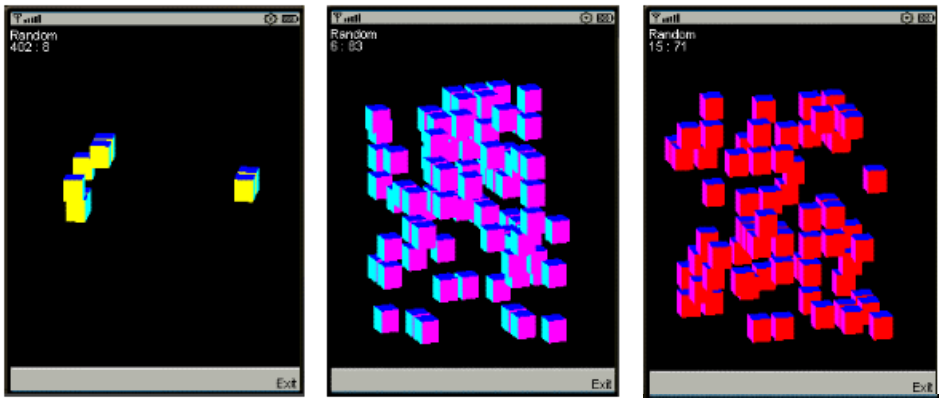

圖 7 IM 模式的繪圖範例

由以上的範例可以得知，IM 模式的繪圖方式比較直接，可以從程式中控制 3D 物件繪製的位置、大小、顏色，甚至可以做到動態產生物件。這在 RM 模式中是無法辦到的。其實在 JSR-184 中，RM 模式與 IN 模式可以共存，因此在製作遊戲上可以採用比較折衷的方式，遊戲場景中不需要變化的部份，可以經由 RM 模式來讀取其 3D 資料，而會變化的部份，如主角就可以使用 IM 的方式來產生。

目前 JSR-184 主要是應用在手機領域，因為當下的 PDA 還未完全支援 Java 虛擬繼器（Java Virtual Machine，簡稱 JVM）。假如 PDA 往後也支援 JVM 的話，相信 JSR-184 在 PDA 的表現也將會相當耀眼。

2.3.3 Direct 3D Mobile 簡介〔26〕

在本節中將會提到由微軟所提出的 3D 繪圖 API。微軟在嵌入式系統的發展是以 PDA 上的操作系統為主，如 PocketPC 系列或是 WinCE 系列。而在目前 3D 潮流的推波助瀾之下，微軟特別針對 PDA 上的 3D 繪圖推出了自家的繪圖 API－Direct3D Mobile。

相信大家都對 Direct3D 這個繪圖 API 都相當的熟習，Direct3D 替微軟在 3D 繪圖的領域中搶下了龍頭的位置，在現今幾乎每台個人電腦都配上微軟的作業系統，因此 Direct 3D 成為了目前最為廣泛使用的繪圖 API。

Direct 3D Mobile 跟 OpenGL ES 類似，也是從 Diret3D 中萃取出來專門針對嵌入式系統所設計。目前在 WinCE5.0 的版本才有支援。Direct3D 有著以下的特色：

1. 針對嵌入式系統的特點做設計，Direct3D Mobile 也只需要少量的資源即可運作。
2. 其驅動架構可以由廠商選擇由軟體驅動、硬體驅動，或是軟體與硬體兩者合作來驅動。

3. Direct3D Mobile 特別針對浮點數作處裡，使其支援使用 16.16 格式的浮點數格式做運算。

不過，當本文在撰寫的同時，WinCE5.0 也還在測試階段，因此目前在市面上也只有微軟提出的 WinCE5.0 遠景中的相關資訊可供參考。就目前來講，Direcr3D Mobile 比起前面兩個 3D 繪圖 API 要較為弱勢，因為目前還無法得知採用 Direct3D Mobile 的廠商的評價為何。不過，以微軟目前在 PDA 上的操作系統的佔有率而言，將來 Direct3D Mobile 的後勢也是相當看好。

2.3.4 在手持式裝置上的 3D 繪圖 API 的比較

在前面三節中，介紹了 OpenGL ES、JSR-184、Direct3D Mobile 三種。下表是以上三種 API 的比較。

表格 2 三種 API 的優缺點比較表

	優　　點	缺　　點
OpenGL ES	1.產業標準和免權利金 2.小容量與低耗電量 3.平台移植性及延伸性佳 4.開發文件齊備	1.不同的硬體規格，會需要找尋適合使用到不同的協定 2.與硬體本身的關連較為緊密
JSR-184	1.為廠商所共同制定規格 2.可以整合 OpenGL ES 3.具有兩種繪圖模式 4.開發可得到的支援豐富 5.平台移植性及延伸性佳	1.在不支援 JVM 以及 JSR184 的機器上，無法使用 2.對於應用程式的限制比較嚴格
Direct3D Mobile	1.WinCE5.0 即支援 2.特別支援浮點數的運算 3.繼承於 Direct3D，開發文件互通	1.非由討論所產生的標準規格，為微軟自家的 API 2.目前尚未有廣泛的應用

經過上表的比較之後，在本論文的實作上，是挑選 JSR-184 作為本研究中系統實作部份所使用的 3D 繪圖 API。

而在第三章，將會對原先 3D 遊戲編輯器的系統進行分析，然後再針對這些不足之處進行修改，最後使得 3D 遊戲編輯器可以支援手機上的 3D 遊戲編輯，達到本論文的研究目的。