銘　傳　大　學

資　訊　工　程　學　系

專 題 研 究 初 審 文 件

本校一一四學年度　資訊工程學系

組員：周孝倫

所提專題研究：點構式向量繪圖系統

指 導 教 授 ：王豐緒

中華民國　一一四　年　十　月　二十八　日

摘要

目前市面上雖已出現各式各樣的向量圖繪圖程式，但幾乎是以點線面架構運作，例如 Adobe 的 illustrator 便是透過這種架構發展，這種架構將無法支持非典型的形狀，缺乏彈性。且如今大多繪圖工具的複雜性越來越高，獨自學習的門檻也越來越大。為了能夠提高初學電繪之學生學習之效率，並提供更加彈性化的開發概念，本研究試圖建立不同於目前主流之點線面架構之向量圖格式，並以使用者操作順暢為宗旨，設計一個架構更加簡單，彈性更加廣泛的物件模型，並使用物件導向技術降低未來增加各式物件種類的成本。

目錄

圖目錄

第一章 緒論

此章將敘述促使本研究產生的動機，以及本研究欲解決的問題。

第一節 研究背景與動機

隨著各式繪圖軟體的推陳出新，繪圖軟體正在往複雜、多功能的方向演化，目前市面上最常用的向量繪圖軟體之一是 Adobe 的 illustrator，此應用程式功能複雜，使其難以獨自上手，通常需有專人指導才能逐步上手。本校商業設計系便有標誌設計、視覺傳達等課程皆須要使用向量圖繪畫，不過對於初學者而言，illustrator 的使用是非常複雜的，這不僅是因為功能龐雜，其中也包括了一些並不直覺的操作方式，例如放大縮小畫面需要按下 alt 鍵並使用滑鼠滾輪，這不像一般通用軟體可直接使用滑鼠滾輪放大畫面，因此在初步接觸時會有明顯的適應不良情況。

並且目前繪圖架構若需增加全新概念的物件，因現有大部分繪圖軟體對物件的定義侷限於點線面，往往需另開新的根類別以滿足需求，導致架構彈性不足與維護成本上升。為了能夠提供初學者一個相對友善且快速上手的繪圖平台，同時提供開發者更加彈性的物件設計架構，本研究的目標在於設計一個以點為核心、簡化形狀定義

的圖形結構，藉此省去繁瑣規則、提升擴充效率與繪圖效能，並降低開發與學習門檻。

第二節 研究問題

我們發現初次使用 illustrator 的學生會有明顯的適應不良情況，這對學習而言是一個自信上的挫敗，而我們推測 illustrator 難以上手的原因有二：

1. 他的常規操作與常見通用軟體常規操作不完全相同。
2. 他擁有更加複雜的功能。

對於更複雜的功能，我們會分別從使用者與開發者的角度去探討如何解決。對於使用者而言，為了能夠維持作品品質，他們依舊需要複雜的功能，但 API 必須精簡，一個工具能夠完成多件任務，才能讓使用者能夠流暢地上手；但對於開發者來說，每一個工具只負責一項任務，才能以低成本維護軟體的各項功能。

而我們發現，目前主流的繪圖軟體都無法逃脫點線面的物件架構，我們希望能夠從根本省略形狀的定義，來達成整體程式簡化的目的，同時提升物件的彈性，讓開發者能夠在一個物件下製作各種子類功能，藉此提供更加輕便、高效率且容易維護的設計模式，並以此為基礎開發出新的向量式繪圖應用程式。

本研究之繪圖應用程式為基於 Java Swing 的繪圖平台，能夠輸出與讀取自定義的檔案格式協定，並支援各種常見之使用者習慣的操作，如快捷鍵全選、存檔、複製、重作、刪除、放大及縮小等等，進一步提升上手的速度。為了驗證本研究架構確實改善現有系統之效能與維護效率，後續將透過架構理解、模組擴充以及相容延展性的分析，來評估本研究方法的有效性。

第二章 概念探討

第一節 向量圖理論

向量圖結構

向量圖是一種基於函數或數學向量所構成之圖形，目前常見的向量圖的檔案格式(.svg)相對於點陣圖，儲存了更輕量的資訊，因此很適合做為大量3D 物件的呈現形式，目前知名的商用向量圖繪圖應用程式如 Adobe illustrator 是付費訂閱制的，該應用程式將向量解讀為線的集合，再將線解讀為點的集合，點分為角點與平滑點二種，分別可做出直線與曲線，對於曲線，大部分繪圖軟體應用的是三次貝茲曲線函數，即

$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$ ，其中$t \in [0,1]$。

目前常見的向量圖的檔案為(.svg)，其檔案結構基於 XML，逐段描述物件，下為一向量圖檔案之描述結構演示：

<circle cx="50" cy="50" r="40" stroke="black" fill="red" />

本段代表宣告一圓心為(50,50)，半徑為40的黑邊紅色圓形，十分直觀，人們可以直接解讀。

實際應用

目前向量圖大範圍應用於3D 應用程式中，例如虛擬引擎 Unity3D，其組件之呈現外觀為頂點向量之集合，這種設計相較於點陣圖，儲存更少的資料，視覺上也更加平滑；此外向量圖同樣應用於商標設計上，因為大部分商家並不會希望自己的商標放大後產生鋸齒的瑕疵，且商標通常不注重寫實，因此也是向量圖經常使用的場合。向量圖有一缺點，就是難以模擬現實的複雜情境，因此目前大部分3D 應用程式與遊戲，都是透過向量方式建構物體的形狀，並使用點陣圖附著在形狀之上，這在 Unity3D 中被稱為材質。

第二節 自創之點構式繪圖邏輯概念

物件的組成

不同於如今對於面的概念，點構式向量繪圖邏輯中，物件僅具備以下三種元素：繪製邏輯、點集合、顏色。由繪製邏輯設計外觀

與行為，如脫拽、中心點定義、放大縮小之定義、移動定義、與點之關係描述、合法存在條件等等。這種架構不僅相對現在的概念少了一個定義，也將物件的彈性擴張到與外界交互的程度。

點的概念

點可分為動態點與靜態點，靜態點的表現為使用者不可直接決定其位置，靜態點的位置與屬性是透過繪製邏輯決定的，它可以設計為相對物件靜止，因此稱之為靜態點。靜態點可做為中心點、凸包的快取、選中範圍判斷依據等等；動態點則被圖形引用數值，是經典圖形的形狀參考，可被外界直接改變，因此稱為動態點。

靜態點與動態點兩概念的設計，使物件不再侷限於形狀，而是具備與其他物件交互之能力之物件，如群組，群組的設計可引入四個靜態點維護群組的邊界，並在繪圖邏輯內覆寫各種行為，達到萬物統一的功能。本專案對於初始的中心點定義為所有點集合的重心位置，若一開發者想描述橢圓形，可引入兩個動態點控制長與寬，以及兩個靜態點，一個做為圓心的快取，一個做為邊界凸包處理的快取，若打算以相機對準物件只需對齊凸包即可，無須重複計算邊界範圍。

繪製邏輯

繪製邏輯可設計成具備交互能力之物件，並且彈性極大，可覆蓋各式行為，這種繪製邏輯更進一步擴張了本邏輯系統之物件概念的彈性，僅需物件類別，即可造出各式物件。

本研究之創新性在於以「點為核心」重構向量繪圖架構，透過靜態點與動態點的分層設計，省略傳統形狀定義所需的中介層，進而提升物件通用性與系統擴充性。

第三節 概念實現之應用程式設計

操作設計

本研究所研發之應用程式是基於 Java Swing 繪圖工具、引用點構式向量繪圖概念的邏輯，程式名為 painter，painter 具備兩種操作模式：應用程式的編輯模式與閱覽模式，應用程式具備多種快捷工具，皆符合通用軟體常見之操作，能夠使初次使用者快速上手，快捷工具如下：

ctrl+C	對選取物件複製
up/right	放大所選取之物件
down/left	縮小所選取之物件
delete	刪除所選取之物件，若無物件選取則刪除最上層圖層
ctrl+Z	重作操作
ctrl+Y	復原操作
ctrl+S	存檔
ctrl+A	全部選取
ctrl+滾輪	視角拉近拉遠
ctrl+滑鼠點擊	選取多重物件
滑鼠左鍵	選取當前最上層之物件或點，或是點空白處取消選取全部物件
滑鼠右鍵	改變所選物件之顏色
檔案托拽	讀取自訂義的檔案格式(.vecf)
視窗左邊	圖層管理器，可滑鼠托拽更蓋物件的圖層順序，可滑鼠托拽邊界改變布局
視窗上面	工具列，可生成出預設圖案，以及群組、解散群組、生成多邊形、多段線、貝茲曲線等，可滑鼠托拽邊界改變布局

圖2-1 輸入操作列表

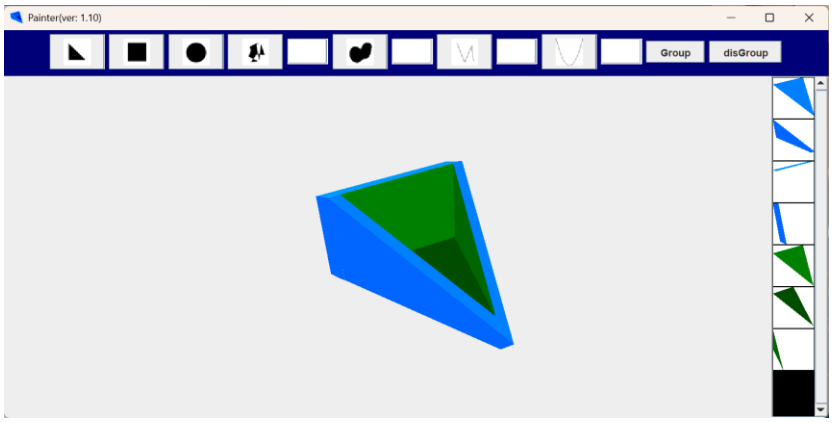


圖2-2 Painter 應用程式畫面演示

自定義的檔案格式協定

本研究為本程式的檔案讀寫設計了單獨的檔案讀取、輸出格式，使用副檔名.vecf，意旨 vector format，在檔案第一行紀錄靈敏度與相機的偏移位置，總共三個浮點數，第二行以後每行都是一個物件描述，其中經典圖案的格式為 [Type_name] [x1] [y1] [x2] [y2] …C [R] [G] [B]，每一個數據之間使用空白間隔。[Type name]是該物件的代號(例如圓形為 Cr)，[x1] [y1]…則為點集合，C [R] [G] [B]是以單字 C 標示後面三個數字 RGB 為顏色。而群組物件則為 G: <Object1> , <Object2>，每個物件用逗號分開，再透過 Object 內宣稱的物件型態去處理<Object>內的資料，生成進群組內。每一個新的物件的類別，都需要覆寫其檔案的儲存格式、繪製邏輯、合法存在條件。

其他功能

本專案支持沒安裝 Java 環境的 Ubuntu 作業系統、Windows 10 以上作業系統，以及各式具有 Java JDK-25 之作業系統上運作，所有平台使用的如果相同版本則都是同一批程式碼，因此大幅保證不同系統更新的同步；本專案提供兩種開啟的入口：閱覽模式以及一般編輯模式。

第三章 可行性分析

本研究以 Java 語言為基礎，並且運用網路上免費的工具支援，才能開發出本次研究之基於點構式向量繪圖系統之應用程式，所需的資源有:Virtual box、github、launch4j、Inno setup 等。

第一節 經濟可行性

本研究所使用的工具有 eclipse、Virtual box、github、hackMD、Inno setup 等

，由於這些都是免費的工具與資源，所以從經濟可行性角度考慮，本研究完全可行。接下來我們將為這些工具做簡單的描述。

eclipse:

eclipse 是由 Eclipse Foundation 所經營，是一個常見的 Java 的免費開發集成環境，具備介面化的操作，能夠省略掉一些步驟的 cmd

指令操作，例如 jar 檔案輸出，並支援快速除錯功能與外部專案導入編輯。

Virtual box:

Virtual box 是由 Oracle 公司經營的虛擬機系統，可透過.iso 檔案模擬不同作業系統的運行，可以用來提供一個安全的隔離操作實驗環境。Virtual box 提供了一個非常方便的多系統支援管道。由於 macOS 的 EULA(使用者合約)規定不可使用非 mac 裝置使用 macOS，因此本研究不支援 macOS 系統。

Github:

Github 是一個專案版本控制的系統，以專案櫃為根基，並支援多人 commit 上船更新，以及隨時恢復舊版的功能，能夠高效率支援跨平台設計的資源同步，以及過去版本歷史的監控，加以確保程式與過去是否能夠兼容。

第二節 技術可行性

本研究所使用的語言只有 Java，Java 可在系上大二必修物件導向，其相關課程中學習，所以從技術可行性角度考慮，本研究完全可行。

第四章 系統分析

此章我們針對應用程式的架構進行分析，其中架構將會使用系統流程圖、樹狀圖與分布圖展示。

第一節 功能分析

此節我們針對應用程式系統的整體架構分析。首先，我們以使用分布圖表示此系統的佈局規劃，接下來會使用樹狀圖，從類別的角度出發，分析類別與類別之間的關係。再利用系統流程圖，個別展示一般應用程式模式與閱覽模式的執行流程。

第一項 使用分布圖

我們使用 java 自帶的 BorderLayout 對所有主動操作的工具做管理，我們設計的可主動操作功能說明如下：

工具欄：

工具欄可以增加各種功能之元件，如新增特定形狀、群組、導入點陣圖檔案，吸管吸取顏色等等。

圖層管理器：

實時將畫布中物件的圖層呈現在管理器內，並且可手動拖動元件更改物件的圖層順序。

畫布：

繪製物件的區域，偵測各種輸入事件，提供直覺化的操作介面。

圖 4-1 分布圖

第二項 使用樹狀圖

我們的核心環境是 Scene 類別，具備整體操作功能的 API，內部類別有 Note 與 ChoiceColor，以及相機的三個變數:scale, offsetX, offsetY 分別代表縮放程度與偏移座標。Note 類別提供三個 API: saveInfo, redo 與 undo，可儲存狀態、恢復與撤銷。

可見層 mainPanel 與 ExportLoadSystem、LayerManager 以及 ToolList 類別關聯，其中 ExportLoadSystem 有兩個內部類別:Loader&Exporter，負責輸入與輸出；LayerManager 有個內部類別 DraggableItem，並且有一個 DraggableItem 的列表紀錄類別關係，並

藉此管理畫布上的元件出現先後順序。代表管理器內偵測畫布上元件的呈現元件，該內部類記錄其所屬的物件；ToolList 有個內部類別為 Tool，他繼承自 JButton，是本程式原生提供的 API，能儲存要做的事情以及在布局中呈現的圖片或字串。

PainterObj 是所有物件如群組、圓形等等的根類別。他儲存點陣列、繪圖流程、操作邏輯、合法存在條件、物件之檔案格式、是否可被拖動等等，所有方法之預設流程皆按照經典的多點控制式形狀設置，同時也提供必要 API，他被 Scene 紀錄入物件列表中，並依照是物件是否可拖動加入可拖動列表中；PainterObj 擁有歸屬的類別 Point 與 Color，Color 儲存 RGB 值；Point 儲存 XY 值以及所屬物件。

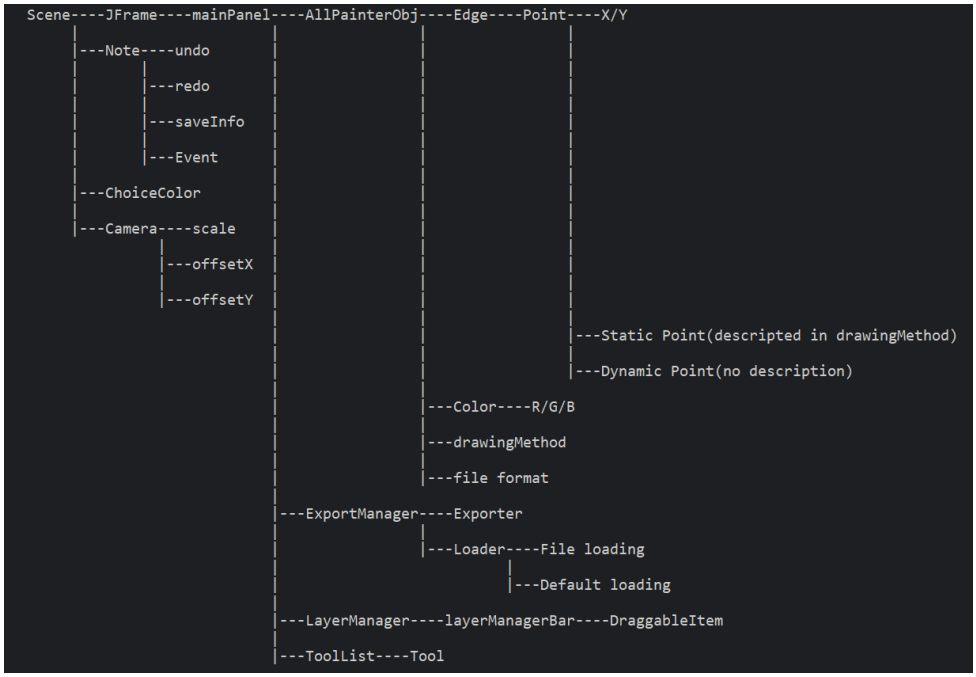


圖4-2 樹狀圖

第三項 使用流程圖

本流程圖將呈現本程式兩個開啟方式(編輯模式以及閱覽模式)的流程。

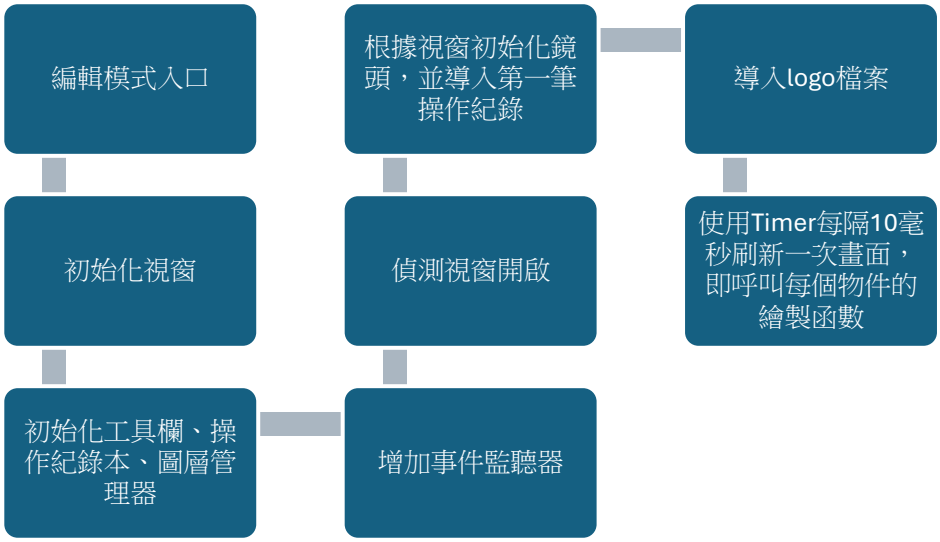


圖4-3 使用流程圖表示編輯模式流程

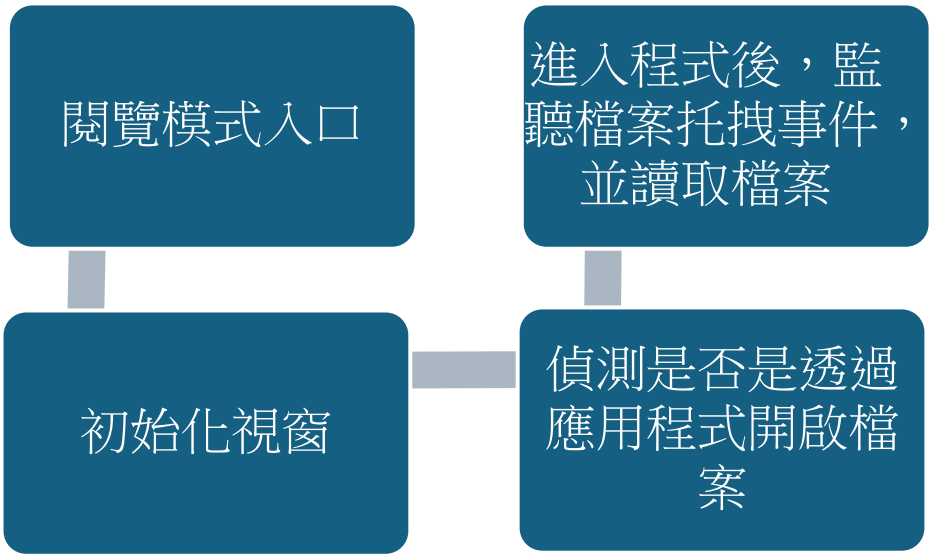


圖4-4 使用流程圖表示閱覽模式流程

第二節 專案分析

此節我們使用樹狀圖來表示專案資料夾的結構，有助於開發者接手以及統一資料夾功能分類。

資料夾樹狀圖

```
Painter:
    bin:
        javafile:
            -compiled Class File
        testing:
            -painterBrowser.class
            -Main.class
    src:
        javafile:
            -ExportManager.java
            -LayerManager.java
            -Scene.java
            -PainterObj.java
            -ToolList.java
        testing:
            -painterBrowser.java
            -Main.java
    resource:
        -painter_logo.ico
        -painter_logo.png
        -triangle.png
        -circle.png
        -Nedge_BL.png
        -Nedge_BS.png
        -Nedge_SL.png
        -Nedge_SS.png
        -quad.png
        -file.txt
```

圖4-5 專案架構

```
Assets:
    v1.7.1~v1.10:
        jar files:
            -painter.jar
            -painterBrowser.jar
        Windows:
            -painter.exe
            -painterBrowser.exe
        Ubuntu:
            -painter:
                bin:
                    -painter
                lib:
                    app:
                    runtime:
                    libapplauncher.so
                    painter.png
            -painterBrowser:
                bin:
                    painterBrowser
                lib:
                    app:
                    runtime:
                    libapplauncher.so
                    painter.png
    -report.docx
    -readme
```

圖4-6 版本控制

第三節 研究方法有效性評估

本節我們將會評估本研究對於開發者擴充功能的成本，我們會分別以架構理解、模組擴充以及相容延展性做評估。

架構理解

本項將會以「是否能在一張 A4 圖中清楚畫出資料從點 → 關係 → 形狀 → 繪出之流程」來判斷架構是否容易理解，如果需要太多註解或跨線交錯、需要標示很多暫存變數、中間類別以及例外處理，代表結構太複雜，下面將會以貝茲曲線構成面的程式碼來分析新增一個新的形狀的流程。

```
class BezierSurface extends PainterObj {
    public BezierSurface(Scene scene) {
        super(scene);
    }
    public static BezierSurface INSTANCE(Scene scene) {
        BezierSurface s=new BezierSurface(scene);
        s.addPoint(0,0);
        s.addPoint(0,1);
        s.addPoint(1,1);
        s.addPoint(1,0);
        return s;
    }
    @Override
    public void draw(Graphics g,double scale,double offsetX,double offsetY) {
        if(islegalObj()) {
        setDrawingColor(g,this);
        Graphics2D g2=(Graphics2D)g;
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setColor(g.getColor());
        Path2D path=new Path2D.Double();
        path.moveTo(this.getEdge()[0].getX()*scale+offsetX,this.getEdge()[0].getY()*scale+offsetY);
        if(this.getEdge().length>3) {
            for(int i=1;i<this.getEdge().length-2;i++)
            path.curveTo(this.getEdge()[i].getX()*scale+offsetX,this.getEdge()[i].getY()*scale+offsetY,this.getEdge()[i+1].getX()*scale+offsetX,this.getEdge()[i+1].getY()*scale+offsetY,this.
        path.curveTo(this.getEdge()[this.getEdge().length-2].getX()*scale+offsetX,this.getEdge()[this.getEdge().length-2].getY()*scale+offsetY,this.getEdge()[this.getEdge().length-1].getX()*
        }
        else {
            path.quadTo(this.getEdge()[1].getX()*scale+offsetX,this.getEdge()[1].getY()*scale+offsetY,this.getEdge()[2].getX()*scale+offsetX,this.getEdge()[2].getY()*scale+offsetY);
            path.quadTo(this.getEdge()[2].getX()*scale+offsetX,this.getEdge()[2].getY()*scale+offsetY,this.getEdge()[0].getX()*scale+offsetX,this.getEdge()[0].getY()*scale+offsetY);
            path.quadTo(this.getEdge()[0].getX()*scale+offsetX,this.getEdge()[0].getY()*scale+offsetY,this.getEdge()[1].getX()*scale+offsetX,this.getEdge()[1].getY()*scale+offsetY);
        }
        path.closePath();
        g2.fill(path);
        }
    }
    @Override
    public String toString() {
        return "BS "+this.DescriptPoint();
    }
}
```

圖4-7 範例經典圖形的繪製流程程式碼

本程式碼具有以下函數：建構子、預設實例、繪畫方式、檔案輸出方式。

預設實例的流程：宣告實例 → 新增點位置 → 回傳實例

繪畫方式的流程：如果當前的結構合法(預設為具備至少 3 個點)則取得顏色，在第一個點的位置下筆，然後遍歷每個點，如果只有三個點則劃出 2 次貝茲曲線，否則使用三次貝茲函數劃出路徑，最終封閉路徑並填充。

模組擴充

本項會以「一個新功能能否在不破壞既有系統下被自然加入」為判斷依據，進行模組擴充速度的評估，我們將打算加入「文字」類別做擴充彈性實驗，下面將會附加新物件程式碼架構的圖片以及成果。

圖4-8 添加非經典圖形物件「文字」的範例圖

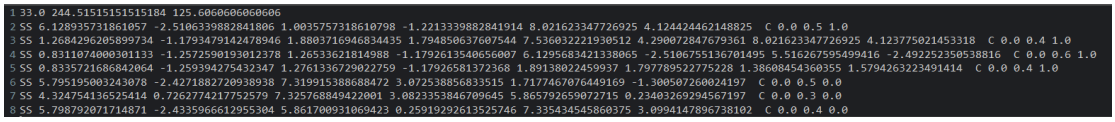
```
class Text extends PainterObj{
    private String text="";
    private double fontSize;
    private double X,Y;
    public Text(Scene scene) {
    }
    public Text(Scene scene,String text,double X,double Y,double fontSize) {
    }
    public String getText() {
    }
    @Override
    public boolean islegalObj() {
    }
    @Override
    public void draw(Graphics g,double scale,double offsetX,double offsetY){
    }
    @Override
    public void loadfile(String[]array) {
    }
    @Override
    public boolean isPointInPainterObj(int mx,int my,double scale,double offsetX,double offsetY) {
    }
    @Override
    public String toString() {
    }
    @Override
    public Text clone() {
    }
    @Override
    public void changeSize(double rol,double cx,double cy) {
    }
    @Override
    public void moveX(double X) {
    }
    @Override
    public void moveY(double Y) {
    }
}
```

圖4-9 非經典物件「文字」的宣告

這項修改非經典圖案，仍能不改變現有架構自然增加，可得知目前架構具備充足的模組擴充能力。

相容延展

本項將會分析由舊程式版本所做出的檔案是否能被新版本的程式所開啟，在這裡我們將使用應用程式的 logo 檔案作評估。



```
1 33.0 244.51515151515184 125.6060606060606
2 SS 6.128935731861057 -2.5106339882841806 1.00357573186107 98 -1.2213339882841914 8.021623347726925 4.124424462148825  C 0.0 0.5 1.0
3 SS 1.2684296205899734 -1.1793479142478946 1.8803716946834435 1.794850637607544 7.536032221930512 4.290072847679361 8.021623347726925 4.123775021453318  C 0.0 0.4 1.0
4 SS 0.8311074000301133 -1.2572590193012378 1.26533621814988 -1.1792613540656007 6.1295683421338065 -2.5106755136701495 5.516267595499416 -2.492252350538816  C 0.0 0.6 1.0
5 SS 0.8335721686842064 -1.259394275432347 1.2761336729022759 -1.17926581372368 1.89138022459937 1.797789522775228 1.38608454360355 1.5794263223491414  C 0.0 0.4 1.0
6 SS 5.795195003243078 -2.427188272093893 8 7.319915388688472 3.072538856833515 1.7177467076449169 -1.300507260024197  C 0.0 0.5 0.0
7 SS 4.324754136525414 0.7262774217752579 7.325768849422001 3.0823353846709645 5.865792659072715 0.2340326929456719 7  C 0.0 0.3 0.0
8 SS 5.798792071714871 -2.4335966612955304 5.861700931069423 0.25919292613525746 7.335434545860375 3.0994147896738102  C 0.0 0.4 0.0
```

圖4-10 logo 的.vecf 檔案內容

此檔案建立於版本1.3，已經歷超過8次的更新，因此可得知本程式對於其舊版具備足夠的兼容性。

第五章 結論

本章節由以下的甘特圖表示當前成果與未來方向

	Ver 1.11	Ver 1.12	Ver 1.13	Ver 2.0	Ver 2.1
優化垂直水平移動機制	已完成				
增加精準輸入操作	製作中				
增加文字物件	已完成				
支援 SVG 檔案					
增加矩陣轉換功能					
增加漸層					
增加自動吸附演算法					
測試 Java 8向下兼容性					
優化演算法提升計算能力					

圖5-1 未來功能甘特圖

當前成果:

本研究當前版本為 1.10，已經完成大部分的使用者習慣功能、物件導向架構成熟，並對點構式向量繪圖概念進行重點整理與分析技術可行性。

未來方向:

本專案未來將會持續降低使用者的使用成本同時擴張功能，以及滿足與當前商用向量圖原理相互轉換的功能，並逐漸提升計算速度。

參考文獻

一、 中文部分

1. 王豐緒（2015）。初審初稿（範例）
https://ms1mcuedu-my.sharepoint.com/:w:/g/personal/mcucsie506_ms1_mcu_edu_tw/Eagm7RLRCelCkm7FNRxq7EMBaC7ibo0TeA7wTfq_NyjWfg?rtime=zyI8FY8V3kg

2. 銘傳大學（2025）。專研專刊—專刊格式。
https://ms1mcuedu-my.sharepoint.com/:w:/g/personal/mcucsie506_ms1_mcu_edu_tw/EfpRsS53w1hElMGHEMvz39QBYQ1zGif5fOH2dR0aZai76g?e=ebL3W0