|  |
| --- |
|  |
| Interactive Point-Controlled Vector System |
| 13360331 資工二甲 周孝倫 |

|  |
| --- |
| 孝倫 周  2025/10/25 |

**Abstract**

This project aims to design and implement an interactive vector drawing system in Java, named **Painter**.  
The core concept of the system is "point-driven," allowing users to directly manipulate control points to modify the shapes of objects, achieving high interactivity in vector graphics editing.  
Unlike traditional drawing software that stores only pixel information, this system is implemented with an object-oriented architecture, where all graphics inherit from the base class PainterObj and are defined by control Points, ensuring a clear structure and high scalability.  
Additionally, the system includes a grouping function, enabling multiple objects to be manipulated as a single group and allowing group decomposition for later adjustments.  
The system also implements a layer manager and file import/export functionality, providing a complete professional drawing workflow.  
The overall design emphasizes modularity, expandability, and user experience.

**Chapter 1: Introduction**

**1.1 Motivation**

Traditional drawing systems mostly operate in a "bitmap mode," which is intuitive but loses resolution when scaling or copying images.  
Vector drawing systems can mathematically describe graphic structures precisely, but existing software is often complex and difficult to learn.  
Therefore, this project aims to develop a lightweight, interactive, object-oriented vector drawing platform suitable for educational and design purposes.

**1.2 Objectives**

1. Implement a drawing system with an object-oriented architecture.
2. Provide an interactive point-controlled editing mode.
3. Support multi-layer management and object grouping.
4. Implement file save and load functionality for project management.

**Chapter 2: System Analysis**

**2.1 System Requirements**

* **Functional Requirements**:
  + Create and delete various shapes (polygons, circles, Bézier surfaces, lines, etc.).
  + Drag control points to modify object shapes.
  + Grouping and ungrouping functions.
  + Layer ordering and visualization.
  + File import/export.
  + Undo/Redo and select-all operations.
* **Non-Functional Requirements**:
  + Simple and intuitive user interface.
  + System stability and scalability.
  + Support future AI-assisted drawing or algorithm integration.

**2.2 System Architecture**

The system consists of the following modules:

| **Module** | **Function** |
| --- | --- |
| PainterObj | Base class for all graphic objects |
| Point | Control point object that determines object shape |
| Scene | Manages all objects, coordinates, scaling, and offsets |
| LayerManager | Manages layer order and drag-and-drop sorting |
| Group | Group functionality for multiple objects |
| ToolList | Toolbar buttons and interactive UI |
| ExportLoadSystem | File saving and loading mechanism |

**Chapter 3: System Design**

**3.1 Class Design**

Based on **object-oriented design principles**:

* Each graphic class (e.g., Surface, Circle, BezierSurface, Line) inherits from PainterObj.
* Point objects maintain a reference to their parent object for bidirectional association.
* Shape behavior methods (draw(), scale(), translate()) are overridden for each graphic type.

**3.2 Data Flow**

1. Users add objects via ToolList.
2. Objects are managed by Scene and displayed in LayerManager.
3. Dragging control points triggers PainterObj to recalculate shape geometry.
4. For grouped objects, operations apply to all children.
5. Scene states are recorded by Note for undo and redo.

**Chapter 4: Implementation**

**4.1 Key Features**

1. **Interactive Control Points**: Users can modify object shapes in real time.
2. **Bézier Curve Editing**: Supports freehand curve drawing.
3. **Grouping**: Multi-object manipulation.
4. **LayerManager**: Visual layer display and ordering.
5. **Auto-Save**: Scene states are automatically recorded after operations.

**4.2 Technical Details**

* Language: Java (JDK 17+)
* UI: Swing
* Graphics: AWT.Graphics
* File format: Custom text format
* Design Patterns:
  + Factory (object creation)
  + Composite (grouping)
  + MVC-inspired layer separation (Scene / Layer / UI)
* **Chapter 5: Testing and Results**

| **Test Item** | **Result** |
| --- | --- |
| Dragging control points to change shapes | Successful |
| Multi-layer object ordering | Successful |
| Grouping and ungrouping | Stable |
| File import/export | Correct scene saving and restoring |
| Large number of objects simultaneously | Stable performance |

**Chapter 6: Conclusion and Future Work**

The system successfully implements an interactive point-controlled vector drawing platform using object-oriented design.  
Operations are smooth, the code structure is clear, and the system is easily expandable.

Future work includes:

1. Implement multi-selection and advanced deformation tools.
2. Integrate AI-assisted point generation or curve smoothing algorithms.
3. Optimize UI to resemble professional vector drawing software.
4. Support layer transparency and style customization.