Aint Paint

Developer Documentation

Log

2-21-14: Created by Kenny Bowers

Introduction

This program allows the user to draw multiple designs on a canvas using the provided tools. Users can control a drawing's individual attributes like line thickness, fill color, etc. Users can save and load the file.

Class Structure

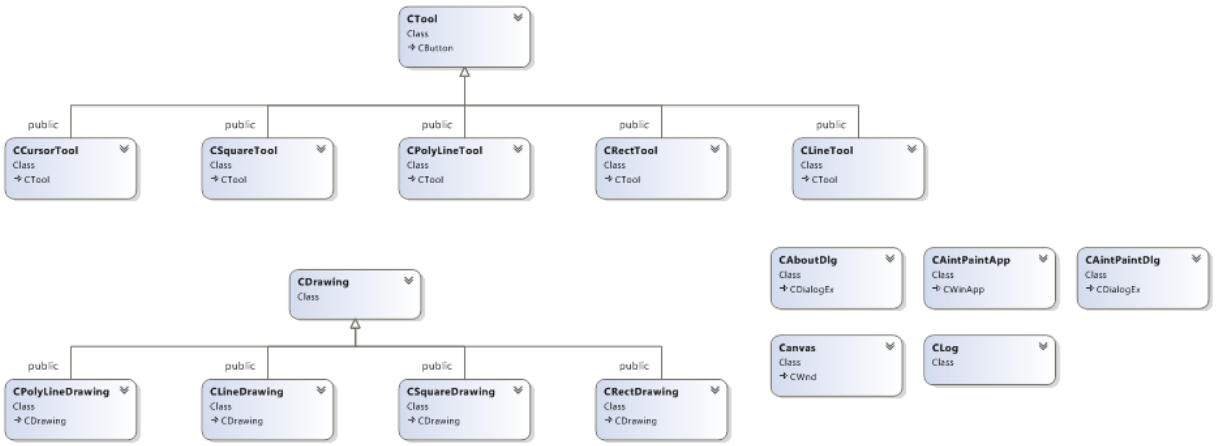

Figure 1. Diagram of Classes and Inheritance.

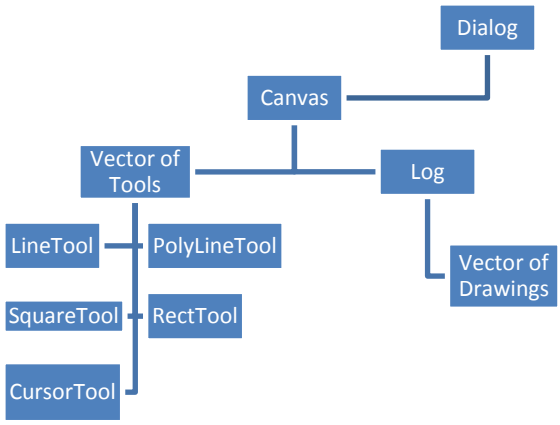

Figure 2. Diagram of Initial Objects.

The Dialog consists of buttons the user can interact with, and also the Canvas. The Canvas is where all drawing occurs. Inside the Canvas object are the tool objects and a log of the current drawings on the Canvas. The Canvas tracks the CurrentTool by Dialog button handlers. A vector contains the Tool objects

which are also enumerated in Dialog. This simplies the communication between the Canvas and Dialog during tool selection. When a mouse event occurs inside the Canvas, it is forwarded to the CurrentTool. After a new drawing occurs, the Canvas is invalidated, which calls the onPaint function. This then clears the Canvas and draws the log of drawings.

There are two abstract classes: CTool and CDrawing. Various types of tools are derived from CTool. Each tool creates its respective drawing, i.e. CSquareTool produces CSquareDrawing objects. Since each Tool handles the Canva's ButtonDown, ButtonUp, DoubleClick, and MouseMove events differently, using virtual functions was a good method to simplify the code.

By allowing CDrawing to be the base class of all Drawings, this allows the Log to easily iterate through a list of different types of Drawings. CDrawing's virtual functions lets each subclass to specify its own drawing algorithms. Drawings also have the ability to search itself to find if a given (x,y) coordinate is located inside it. This was useful for the Cursor Tool's hit detection. Each drawing has an ID which allows for unique identificaiton in the log. CDrawing has a public vector of (x,y) points when private would be preferred, but problems in the development of the save/load functionality required this quick fix.

All the Tools produce Drawings, except for the Cursor Tool. The Cursor Tool reads a user's Canvas click and searches the log to see if the user has clicked a drawing. When a new thickness or color is selected, this is updated in each tool. When the Cursor Tool's settings are updated, it checks if any drawing has been selected, and if so, it changes the settings of that drawing as well.

The CLog class manages a list of CDrawings which is used to redraw the screen and save the file. The log is drawn from the top of the log to the bottom. A new drawing is appended to the end of the log. The log also provides the function to bring a selected drawing to the front of the Canvas. This is done by simply finding the selected drawing in the log, and then moving it to the end of the log. When a drawing needs to be modified, the log is searched for the drawing and the attributes are updated.

Class Functionality

```
class CLog
{
public:
	CLog();
	~CLog();
	void Append(CDrawing * pDrawing);
```

- Reads in a pointer to a Drawing and appends the Drawing to the end of the log vector

```
	void DrawAll(CDC& dc);
```

- Iterates through the vector and draws each drawing

```
	CDrawing * FindXYinLine(CPoint point);
```

- Searches each drawing in the log for the given point

```
	int BringtoFront(int DrawingID);
```

- Finds the given DrawingID in the log vector and moves it to the end of the vector

```
	int SaveToFile(std::string FileName);
```

- Opens a new text file and iterates through each drawing in the vector and saves the drawing's settings to the text file
- <LINETYPE LINETHICKNESS LINECOLOR FILLCOLOR p1.x p1.y p2.x p2.y>
- i.e. Line 3 FFFFFF FF8000 0 0 50 50

```
	int Clear();
```

- Deletes the entire log
- Also used when loading a new file

```
private:
	std::vector<CDrawing *> vpDrawings;
};
```

```
class Canvas : public CWnd
{
	DECLARE_DYNAMIC(Canvas)
public:
	Canvas();
	virtual ~Canvas();
	CPoint startpt, endpt;
	afx_msg void OnPaint();
		• Sets the device context and prints the log
	afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
		• Triggers the Current Tool's ButtonDown event and Invalidates screen
	afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
		• Triggers the Current Tool's ButtonUp event and Invalidates screen
	afx_msg void OnMouseMove(UINT nFlags, CPoint point);
		• Triggers the Current Tool's MouseMove event and Invalidates screen
	int SetTool(int newTool);
		• Sets the Current Tool to the tool object in the tool vector
		• Ends the previous drawing if still in progress
	int BringtoFront();
		• Searches the log to bring the selected drawing to the front of the canvas
	afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
		• Used to end a PolyLine drawing
	void SetLineColor(COLORREF color);
		• Iterates through the tool vector to set each tool's settings
	void SetFillColor(COLORREF color);
		• Iterates through the tool vector to set each tool's settings
	void SetThickness(int Thickness);
		• Iterates through the tool vector to set each tool's settings
	void SaveAs();
		• Calls the log's save function
	void Load();
		• Clears the log and reads the text file line by line
		• Stringstream is used to divide the line based on spaces
		• Each value is read and creates a new Drawing accordingly
		• Appends new Drawing to the log
	afx_msg BOOL OnEraseBkgnd(CDC* pDC);
protected:
	DECLARE_MESSAGE_MAP()
private:
	CTool * pcurrentTool = NULL;
	CLineTool LineTool;
	CPolyLineTool PolyLineTool;
	CSquareTool SquareTool;
	CCursorTool CursorTool;
	CRectTool RectTool;
	CLog * pLog = new CLog;
	FILE * DrawingTxtFile;
	std::vector<CTool*> vtoolList;
	enum TOKENS { DRAWINGTYPE, LINETHICKNESS, LINECOLOR, FILLCOLOR, POINTX, POINTY,
NUM_TOKENS };
};
```

```
class CDrawing
{
public:
	CDrawing();
	~CDrawing();
	virtual void Draw(CDC& dc) = 0;
```
- Draws the appropriate subclass' shape to the given device context (canvas)

```
	virtual int FindXYinLine(CPoint point) = 0;
```
- Uses geometry to determine if a point is located in a drawing within a set threshold

```
	int getID();
	virtual int SetLineColor(COLORREF color);
	COLORREF GetLineColor();
	virtual int SetFillColor(COLORREF color);
	COLORREF GetFillColor();
	virtual int SetThickness(int Thickness);
	COLORREF GetThickness();
	std::string GetType();
	int SetType(std::string Type);
	virtual void setEndpt(CPoint point);
	std::vector<CPoint> vXYpoints;
```
- Vector containing the (x,y) points related to drawing the object

```
private:
	int thickness;
	COLORREF fillColor;
	COLORREF lineColor;
	std::string DrawingType;
	int ID = rand() % 5000;
};



class CTool :
	public CButton
{
public:
	CTool();
	~CTool();
	virtual void onDownClick(CPoint point, CLog * pLog) = 0;
	virtual void onUpClick(CPoint point, CLog * pLog) = 0;
	virtual void onMouseMove(CPoint point, CLog * pLog) = 0;
	virtual int BringtoFront(CLog * pLog) = 0;
	virtual void onDoubleClick(CPoint point, CLog * pLog) = 0;
	virtual void EndDrawing() = 0;
	virtual int SetLineColor(COLORREF color);
	COLORREF GetLineColor();
	virtual int SetFillColor(COLORREF color);
	COLORREF GetFillColor();
	virtual int SetThickness(int Thickness);
	COLORREF GetThickness();
	DECLARE_MESSAGE_MAP()
private:
	// Default Settings
	COLORREF lineColor = 0;
	COLORREF fillColor = 0;
	int thickness = 3;
};
```