

1. Aufgabenstellung

Im folgenden wird die Aufgabenstellung zur Entwicklung eines objektorientierten Grafikprogrammes formuliert; die Aufgabenstellung ist zweigeteilt in die Basisaufgabe und die **erweiterte Aufgabe**. Es empfiehlt sich, zu Beginn gezielt nur die Basisaufgabe zu verfolgen. Die erweiterte Aufgabe kann bei sauberem objektorientierten Entwurf leicht im nachhinein ergänzt werden.

Das Programm soll folgende Grafikobjekte interaktiv **erzeugen** können und die **nachträgliche Veränderung** erlauben:

- Punkte
- Gerade Linien
- Rechtecke
- Ellipsen
- Rechtecke mit abgerundeten Ecken

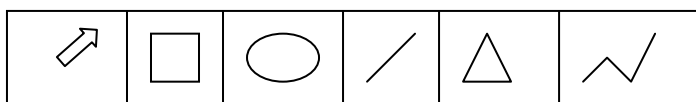
Bei erweiterter Aufgabenstellung zusätzlich:

- Dreiecke
- Freihandlinien (Zusammensetzung aus vielen Punkten)
- Polygonenzüge

Punkte und Linien besitzen eine frei wählbare Farbe und eine wählbare Liniendicke; (*frei wählbarer Liniensstil wie gestrichelt unterstützt nur wenn Dicke von 1 / muss nicht!*)

alle Flächen besitzen eine Farbe und Linienstärke für die Umrandung, sowie eine Farbe, mit der die umrandete Fläche gefüllt (*nur komplett*) wird.

(wichtigste Menüpunkte als toolbar-buttons;



Es gibt immer eine aktuelle Liniendicke, Linienfarbe und Füllfarbe.)

Alle gezeichneten Objekte können markiert werden und wechseln bei Markierung sofort in den Vordergrund. Ein markiertes Objekt wird anders dargestellt (z.B. inverse Farben oder mit anderem Liniensstil); es kann beliebig verschoben werden. Ein markiertes Objekt kann gelöscht werden.

Bei erweiterter Aufgabenstellung zusätzlich:

Über einen Klick mit der rechten Maustaste gelangt man bei einem markierten Objekt in ein Kontextmenü, über das die Objekteigenschaften (Farben und Linien) veränderbar sind.

In der Statuszeile werden jederzeit die x- und y-Koordinaten der aktuellen Mausposition angezeigt.

2. Implementierungsvorgaben

2.1 Benutzerschnittstellen

In einem Menü „Darstellung“ gibt es Menüpunkte zur Wahl der aktuellen Linienfarbe, Linienstärke und Füllfarbe. Die Farben werden über einen Standardfarbdialog ausgewählt.

In einem Menü Zeichnen werden die verschiedenen Zeichenoperationen (Linien zeichnen, Rechtecke zeichnen, ...) angeboten. Bitte das Markieren („Zeigen“) als spezielle Zeichenoperation interpretieren.

Zu diesen Menüpunkten gibt es entsprechende Toolbarbuttons.

Bei erweiterter Aufgabenstellung zusätzlich:

In einem Menü „Arbeitsbereich“ gibt es Einträge zur Wahl des Koordinatensystems (Fensterkoordinaten oder Bildschirmkoordinaten) der Darstellung der x/y-Koordinaten der aktuellen Mausposition sowie zum Umschalten der Darstellung zu einem Fadenkreuz (mit der aktuellen Mausposition als Zentrum).

Darüber hinaus lässt sich ein Rastergitter mit frei wählbarem Rasterabstand ein- und ausblenden.

2.2 Anwendung und verwendete Klassen

Die Anwendung kann nach Wahl als SDI- oder MDI-Anwendung realisiert werden.

Eine Klasse CFigur (abgeleitet von CObject /*Basisklasse der MfC*) und davon abgeleitete Klassen CPunkt, CKreis, CStrecke, CRechteck... beschreiben die Grafikfiguren.

In CFigur : Datamember für Füllfarbe (COLORREF [4Byte wert, für rot, blau..])

Zur Datenhaltung:

Die Datenhaltung erfolgt NICHT in der Dokumentklasse, sondern in der Viewklasse. Diese erhält eine verkettete Liste (CObList) als Datamember; Elemente der Liste sind Zeiger auf Figuren, die ihrerseits dynamisch im Freispeicher erzeugt werden.

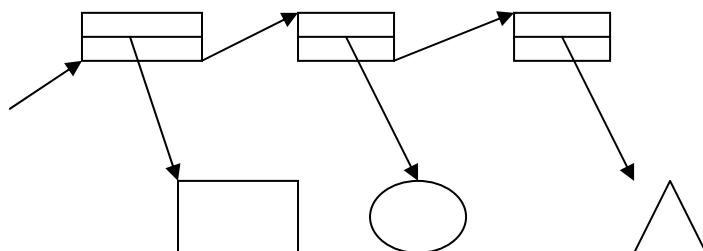
(wenn Programm erweitert werden soll, mit Jochen reden!!)

Verkettete Liste besteht aus einem Zeiger, dieser zeigt auf ein erstes

Listenelement(Verwaltungsinfo, Nettoinfo), dieses auf das nächste Listenelement,

Reihenfolge wird über die Zeigerverknüpfung aufgebaut –

Listenelemente selbst sind Zeiger, die auf die Objekte zeigen



3. Ratschläge zur Implementierung

- 3.1 Festlegung der Dokumentationsform und der Projektgestaltung, hier insbesondere Vereinbarungen über Test- und Releaseprojekte
→ *Benutzerhandbuch (externe Dokumentation)*
- 3.2 Klassenentwurf und Festlegung der Schnittstellen
Die detaillierte Ausarbeitung der einzelnen Klassen kann (und sollte) später erfolgen.
- 3.3 Projekterstellung
Das Releaseprojekt sollte von Anfang an alle später benötigten Klassen einbinden.
- 3.4 Implementierung der Benutzerschnittstellen (Menüs, Toolbar)
Das Programm merkt sich zu jedem Zeitpunkt eine aktuelle Linienfarbe, Liniendicke sowie Füllfarbe und eventuell andere Darstellungsoptionen.
- 3.5 Implementierung der Programmlogik (-Steuerung)
Das Programm befindet sich zu jedem Zeitpunkt in einem der folgenden Zustände (festgehalten in einem Datamember `m_nAktionsModus`):
- Zeigen/Markieren
- Objekt erzeugen (genauer: Rechteck erzeugen, Linie erzeugen, ...)
- Objekt verändern
Die Nachrichtenhandler für `WM_LBUTTONDOWN`, `WM_LBUTTONUP`, `WM_MOUSEMOVE` reagieren auf den aktuellen Aktionsmodus.
- 3.6 Implementierung der Datenhaltung
In einer verketteten Liste, etwa `m_oblistGrafik`, werden Zeiger auf die erzeugten Grafikobjekte gehalten. Die Ausgabefunktion `OnDraw` arbeitet diese Liste von Anfang bis Ende ab und aktiviert für jedes Listenelement die virtuelle Ausgabemethode `ZeichneDich`.
Bei Druck auf die linke Maustaste wird entsprechend dem aktuellen Arbeitsmodus ein neues Objekt erzeugt mit der aktuellen Mausposition als linke obere Ecke und in die Liste als letztes Element eingetragen.
Mausbewegung mit gedrückter linker Maustaste nach Objekterzeugung verändert die rechte untere Ecke der Objektposition (und damit verändert sich das aktuell erzeugte Objekt).
- 3.7 Implementierung EINER Grafikfigur
- 3.8 Schrittweise Implementierung der restlichen Grafikfiguren
- 3.9 Fertigstellung der parallel mitgeführten Dokumentation
- 3.10 [Implementierung der Erweiterten Aufgabenstellung](#)