

Die Komponentensammlung für die Erstellung  
von Grafikanwendungen mit allen wesentlichen Komponenten  
zur erfolgreichen Grafikprogrammierung

**mN** *CadBuilder*

3d cad components collection

## INHALT

3 Daten

4 Device

6 Benachrichtigung (Notification)

7 Interaktivität



## Einleitung

*MNCadBuilder ist eine Komponentensammlung für die Erstellung von Grafikanwendungen.*

*Es werden alle wesentlichen Komponenten zur Verfügung gestellt, die für eine erfolgreiche Grafikprogrammierung notwendig sind. So wird eine mächtige Graphicengine angeboten, die sich wesentlich auf SGI's OpenGL stützt und deren Funktionalität in einer Komponente gekapselt ist. Weiters enthält der MNCadBuilder komfortable Mechanismen zur Implementatierung von Interaktivität.*

*Voraussetzung ist ein Windows Betriebssystem (Windows 98, ME, NT, 2000, XP), und SGI's OpenGL ab Version 1.1.*

*MNCadBuilder ist erhältlich für die Entwicklungsumgebung Delphi der Firma Borland, sowie für das Microsoft .NET Framework.*

# Daten

## Stream

Die Streamingfunktionalität stützt sich auf zwei Komponenten, einem sogenannten *TMNReader* und einem *TMNWriter*. Mit Hilfe dieser Komponenten werden die Daten gelesen bzw. geschrieben.

Es gibt folgende Instanzen von *TMNReader* und *TMNWriter*:

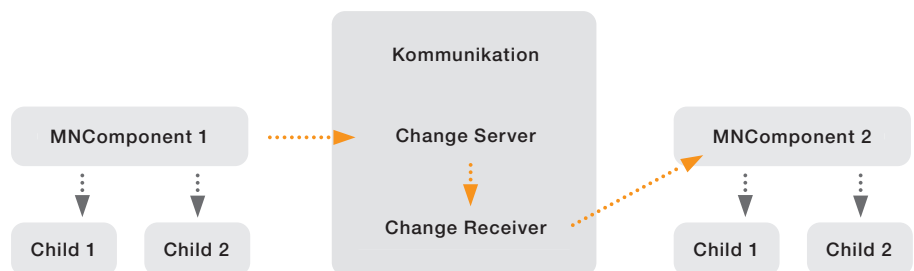
- *TMNAsciiReader* - *TMNAsciiWriter*
- *TMNXmlReader* - *TMNXmlWriter*

Mit Hilfe dieser Reader und Writer ist es möglich, Daten in einem AsciiFormat bzw. XMLFormat darzustellen.

## TMNComponent

Die Klasse *TMNComponent* ist die grundlegende Klasse für fast alle *MNCadBuilder*-Klassen.

Die Kommunikation zwischen den Komponenten findet über einen *Change-Server* und *Change Receiver* statt.



Die drei wichtigsten Aspekte sind:

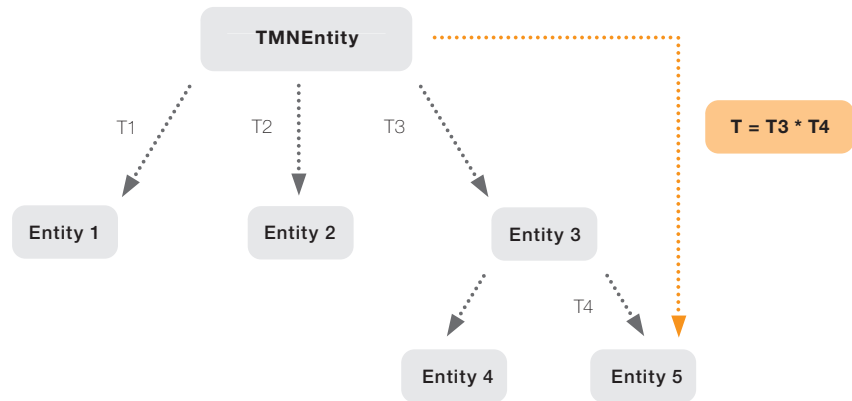
- *Anbinden an das Streamingsystem* Es gibt eine einfache Möglichkeit, eine Komponente vom Typ *TMNComponent* in einen *TMNWriter* zu schreiben bzw. aus einem *TMNReader* zu lesen.
- *Benachrichtigung* Es ist möglich, eine *TMNComponent* mit einer anderen zu „verbinden“, von der Ereignisse wie Löschen oder Ändern mitverfolgt werden und darauf reagiert werden kann.
- *Parentship* Jede *TMNComponent* besitzt einen Parent und eventuell Kinder, sodass durch *TMNComponent* Komponenten eine Baumstruktur erzeugt werden kann.

## TMNEntity

Jedes Entity besitzt eine Transformation relativ zum *MNParent*.

Die Klasse *TMNEntity* ist eine Instanz von *TMNComponent*. Die wesentliche Aufgabe ist die grafische Darstellung. So besitzt jedes *TMNEntity* eine Zeichenmethode, in der die primitiven Zeichenoperationen implementiert sind. Weiters gehört zu jedem *TMNEntity* eine Transformation, der alle Zeichenoperationen unterworfen sind.

Dieser Transformation unterliegen aber auch allfällige Kinder in einer Baumstruktur. Dabei werden die entsprechenden Transformationsmatrizen multipliziert. In einer Baumstruktur, die aus *TMNEntity* Komponenten aufgebaut ist, besitzt somit jeder Knoten eine Transformation und durch Änderung dieser Transformation werden auch alle darunterliegenden Knoten (die Kinder) mittransformiert.



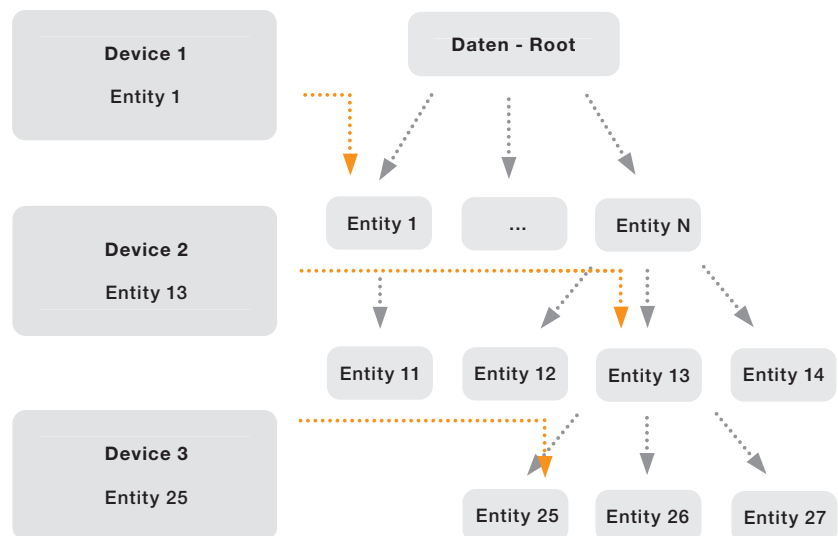
### TMNControl

Neben den Eigenschaften, die *TMNControl* von *TMNEntity* erbt, besitzt *TMNControl* Mausereignisse wie *MouseOver*, *MouseOut*, *MouseDown* oder *MouseMove*. Somit kann ein *TMNControl* verwendet werden, um eine einfache interaktive Funktion auszuführen.

## Device

### Ausgabe von grafischen Objekten

Die Hauptaufgabe der *Device* ist die Ausgabe von grafischen Objekten.



Dies kann auf zwei Arten erfolgen:

- Eine *Device* besitzt ein *OnPaint* - Ereignis. Dieses kann verwendet werden, um primitive Zeichenbefehle zu implementieren. Diese Zeichenbefehle sind Methoden der *Device*.
- Ein *Entity* kann in einer *Device* als *Rootobjekt* gesetzt werden. In diesem Fall wird das *Entity* mit samt seinen Kindern dargestellt. Da in einer anderen *Device* ebenfalls dieses *Entity* als *Rootobjekt* gesetzt werden kann, ist somit eine *MultiDevice* - Fähigkeit gegeben.

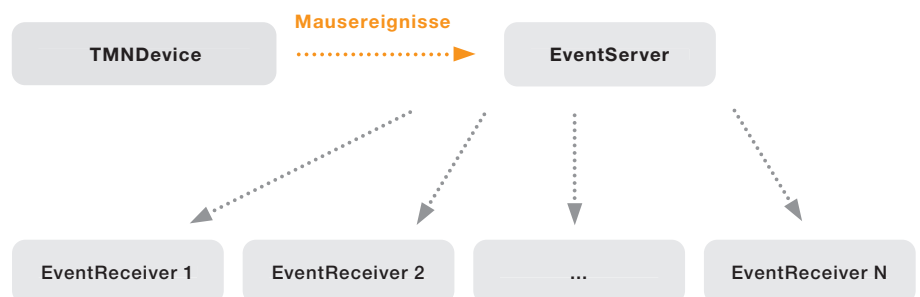
## Windowhandle

Um eine Ausgabe in ein Fenster zu leiten, ist es lediglich notwendig, dessen *Windowhandle* einer *Device* zuzuordnen. Die *Device* besitzt eine eigene Fensterprozedur (*WndProc*), sodass auf Botschaften des Fensters reagiert werden kann. Standardbotschaften wie *WM\_Paint*, *WM\_Resize* usw. werden in der *WndProc* behandelt.

## Mausereignisse

Wenn in einer *Device* Mausereignisse eintreten, so werden diese weiter an einen zentralen *EventServer* geschickt. Verbindet man einen *EventReceiver* mit diesem Server (Siehe *Server – Receiver Prinzip*), so ist es möglich, von jeder Stelle des Programmes Mausereignisse zu empfangen.

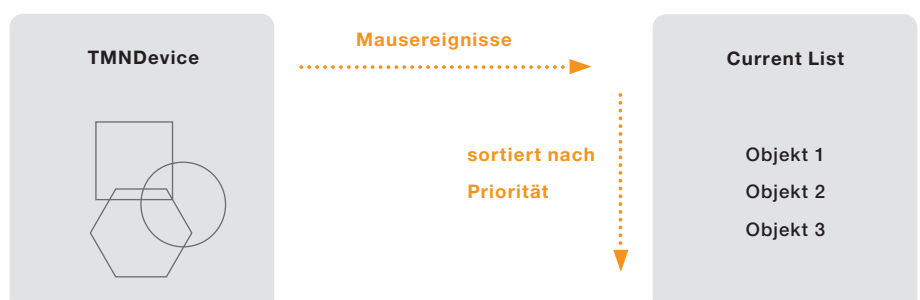
Eine *Device* informiert den *EventServer* über Mausereignisse. Der *EventServer* teilt dann diese Information an seine „angemeldeten“ *Receiver* weiter.



## Select-Mechanismen

In einer *Device* ist der *Select*-Mechanismus implementiert. Selektieren bedeutet, Informationen über jene grafischen Objekte zu erhalten, die an einer bestimmten Position liegen.

Der *Select*-Mechanismus liefert die selektierten Objekte und deren wichtigste grafischen Informationen in der *CurrentList* ab.



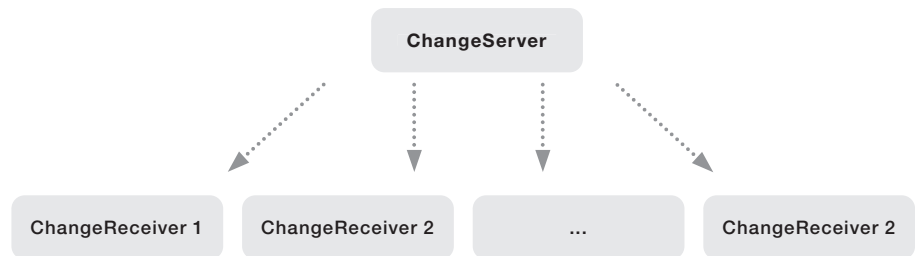
Diese Aufgabe ist sehr aufwendig und so wurde ein eigenes Objekt, der sogenannte *Locator* entwickelt, der den *Select*-Mechanismus weiter unterstützt. Er führt bei jedem *Mousemove* eine Selektion an der Mausposition aus und liefert eine detaillierte Liste von Einträgen, die die selektierten Objekte betreffen.

Diese Liste ist nach einem Prioritätssystem geordnet. Dabei ist ein benutzerdefiniertes Prioritätssystem ebenfalls möglich.

## Benachrichtigung Notification

### Server - Receiver Prinzip

Server sind Komponenten die eine Liste von Receivern verwalten, an die sie Botschaften weiterleiten.



### Implementierte Server

Ein Server leitet Ereignisse an die angemeldeten „Receiver“ weiter.

- *EventServer* Er verteilt Mausereignisse an seine Receiver.
- *ChangeServer* Er teilt seinen Receivern ein *BeginChange*, *EndChange* und ein *Destroy* mit. Jede Komponente vom Typ *TMNComponent* besitzt einen *ChangeServer* und einen *ChangeReceiver*. Veränderungen und Löschen der *TMNComponent* werden von diesem Server versendet. Damit ist es möglich, Benachrichtigungen aufzubauen, in der sowohl auf eine Veränderung als auch auf das Löschen der Komponente reagiert wird.
- *CoordServer* Er teilt seinen Receivern mit, wenn sich Koordinaten verändert haben. Bedient wird er vom *Locator*, der die Koordinaten unter Umständen mit einem Magnetismus auf selektierte Objekte berechnet.
- *MessageServer* Er teilt seinen Receivern eine Botschaft mit, die durch einen *Tag* gekennzeichnet ist. So wird etwa der *MessageServer* bedient, wenn die Eigenschaft *Parent* einer *TMNComponent* verändert wird. In diesem Fall kann an einer anderen Stelle, z.B. in einem Explorer der die grafische Struktur baumartig darstellt, auf die Veränderung der *Parentship* reagiert werden.

# Interaktivität

## Controlobjekte (TMNControl)

Da Komponenten vom Typ *TMNControl* Mausereignisse behandeln können, ist es möglich, einfache interaktive Aktionen zu implementieren.

## Editoren

Für komplexe interaktive Aufgaben wird ein Editor vom Typ *TMNCustomEditor* zur Verfügung gestellt. Er liefert komfortable Grundlagen für eine interaktive Benutzerführung.

Die wichtigsten Bereiche die von ihm abgedeckt werden, sind:

- Datenmanagement mit Markierungsliste
- Prozessmanagement
- graphische Visualisierung von markierten Daten
- Visualisierung der Daten durch eine Eigenschaftsseite (PropertyPage)
- Automatisches *Undo/Redo* System
- Implementierung von Standardfunktionen wie *Copy, Paste, Load, Save, Transform, ...*

## Eigenschaftsseiten (PropertyPages)

Neben der grafischen Visualisierung in einer *Device* gibt es die Möglichkeit der Darstellung durch eine *PropertyPage*. Diese kann verwendet werden um Daten in Textform zu visualisieren und mit der Tastatur zu verändern. Die Daten, die in einer *PropertyPage* und gleichzeitig in einer *Device* dargestellt werden, müssen immer identisch sein. Dies wird durch das *ChangeServer - ChangeReceiver* Prinzips gewährleistet.

# Anwendung

## Zielgruppen

- Software-Entwickler im Grafik/CAD-Bereich
- Software-Entwickler im Allgemeinen, die ihr bestehendes Produkt um die Darstellung von 2D/3D erweitern oder aufwerten wollen

## Beispiele für Anwendungsgebiete

- Zusatzapplikationen im Bereich
  - \_ Maschinenbau
  - \_ Architektur
  - \_ Grafikdesign (Schriften,...)
  - \_ Animation
  - \_ virtuelle Realität im Bereich 3D Spiele
  - \_ Darstellung von gescannten Daten aus Medizin/Naturwissenschaft/Technik/Topographie
  - \_ uvm.
- Aufmass - Software (Aufmessen und Darstellung von Flächen/Volumina)
- Visualisierung von
  - \_ Datenbankeinträgen in 2D und 3D Form
  - \_ Handwerkerlösungen
    - Rohrleitungssysteme visualisieren (Elektriker, Installateure, Spengler, Klimaanlage,...)
    - Bodenleger (zu legende Platten visualisieren, Materialberechnung)
    - Fliesenleger (zu legende Fliesen visualisieren, Materialberechnung)
    - Bau von Messeständen
    - Stiegenbauer
- Profildaten visualisieren und bearbeiten

