3D Algorithmenanimation & Implementierung

Dozent : Michael Ohlmann

Ziel : Verdeutlichung der Notwendigkeit von
3D Algorithmenanimation (Darstellung)
gegenüber (klassischer) 2D Algorithmenanimation

I. Einleitung

- Ursprüngliche Vorgehensweise zum Verständnis eines Programmes (Prozesses) :
  - untersuche Source-Code
  - Debugging

  ⇒ statische 2D-Bilder (Aufrufgraph, Struktogramme)

- Verständniserleichterung durch Verwendung von 2D-Animationen (Abläufen)

- Übergang zu 3D-Animationen (Abläufen), weil :
  - Abläufe an sich mit Datenstrukturen „hantieren“, deren Information 3D-Charakter besitzt (z.B. Matrizen)
  - durch die zusätzliche Dimension mehr Informationen erfaßt werden, die zum Verständnis des Problems notwendig sind.

II. Unterschiedliche Typen von 3D-Darstellungen

Unterscheide 3 Klassen :

1) ***Übertragene 2D-Darstellungen***:
   - keine zusätzliche Information vorhanden
   - dient lediglich ästhetischen Zwecken
     (unter Umständen schnelleres Erfassen durch den Menschen möglich)

   Bsp : Bild 1

2) ***Um zusätzliche Information erweiterte 2D-Darstellungen***:
   - neue Einblicke/Erkenntnisse in den Ablauf möglich

   Bsp : Bild 2

3) ***Darstellungen von Abläufen, die 3D-Charakter besitzen***:
   - Informationsverlust (keine geeignete Darstellung möglich) bei 2D-Darstellung

   Bsp : Bild 3

III. Umsetzung von 3D-Darstellungen (z.B. Polka-3D)

Frage : Warum ein neues Tool, wenn es schon so viele 3D-Animationsprogramme gibt ?
(z.B. PHIGS+, GL, RenderMan)

→
- **Einarbeitungszeit**
- **Wissen aus unterschiedlichen Bereichen**
- **Funktionalität**

Verwendung eines Tools, das :

- einfach zu bedienen ist
- dessen Funktionalität sich im Wesentlichen deckt mit :
  1) **Erzeugung graphischer Objekte**
     (einschließlich Linien, Rechtecke, Kreise, Polygone, Pfeile, Blöcke, Kugeln und Text)
  2) **Dynamische Zuweisung von Position, Größe, Farbe und Sichtbarkeit**
     → Animation
  3) **Darstellung muß äußere Einflüsse berücksichtigen können** (z.B. Änderung der Perspektive zur Laufzeit)
     → Es handelt sich hier nicht um „statische“ Animationen (Animationen, die nach einem fest vorgegebenen Skript ablaufen), sondern um parametrisierte Animationen

→ Tool, das obigen Anforderungen gerecht wird : z.B. Polka-3D
(C++,SGI,GL)

<u>Erstellung einer Animation unter Polka-3D</u>

<u>1.Phase</u> **: Festlegung der darzustellenden Objekte und deren Zuordnung zum Programm**

→ Programmierer identifizieren parametrisierbare Ereignisse (IE), benennen diese, und fügen diese Namen innerhalb des Programmcodes ein (ähnlich Zeus). (Diese Ereignisse sind als Prozeduraufrufe einer Animationskontrollroutine implementiert)

<u>2.Phase</u> **: Entwicklung der Animation**

Aufbau von Polka-3D (Bild 4)

• besteht aus Klassen, die :

1) die gesamte Animation modellieren

- Kapselung der Algorithmenanimation
- Routine der AnimatorKlasse setzt Events um
- Zuordnung Event / Darstellung

2) individuelle Blicke auf die Animation erlauben

- unterschiedliche Darstellungen → Views
- View besteht aus Szenen + View Data Members
- View enthält integer-time-Variable

3) Instanzen, die bei der Erstellung der Ansicht helfen

- Festlegung der Imagery und Aktionen auf dieser innerhalb des Views
- Verwendung von Objekten aus den 3 Klassen (Location,Action,AnimObjects)

- **Location** : - Zuweisung an Objekte aus *AnimObjects*
  - definiert Objekterzeugung und Endpunkte von Bewegungen

- **Action** : - Änderung der Position, Größe, Farbe, Sichtbarkeit von Objekten aus *AnimObjects*
  (KeyWords : MOVE,RESIZE,COLOR)

- **AnimObjects** : - besteht aus Subklassen für 2D und 3D
  - wichtige Memberfunktion : *Program*
    → bindet Objekte aus *Action* an Objekte aus*AnimObjects*
    → verwendet 2 Parameter :
    die verwendete Aktion + time-member-Variable

Die View-Methode ***Animate*** dient schließlich der Erstellung der Animations-Frames und berücksichtigt dabei die time-member-Variable.

<u>Beispielprogramm</u> (Bild 5) :

<u>Weitere Features von Polka-3D</u> :

- Oberflächeneigenschaften (Farbe,Textur)
- Lichtmodell
- Änderung der Sichtposition zur Laufzeit (Kamera)

IV. Beispiele für die Klasse 2 von 3D-Darstellungen

(um zusätzliche Informationen erweiterte 2D-Darstellungen – mit Zeus erstellt)

1) Kürzeste Wege (Shortest Path)

Aufgabe : Bei gegebenem gerichteten Graphen G bestimme ausgehend von einer Quelle q die kürzesten Wege zu allen anderen Knoten.

Algorith :
- Jedem Knoten werden Kosten zugeordnet, und zwar die Kosten des bis dahin gefundenen Weges
- Am Anfang besitzt jeder Knoten bis auf q Kosten unendlich.
- Dann wird rekursiv eine Kante e=(u,v) ausgewählt, und falls cost(v) > cost(u) + weight(e), bekommt der Knoten v die Kosten cost(u) + weight(e) zugeordnet.

Visualisierung (Bild 6) :

2) **<u>Das am nahesten beieinander liegende Paar</u> (Closest Pair)**

<u>Aufgabe</u> : Finde in einer Menge von n Knoten die beiden Knoten, die am nahesten beieinander liegen.

<u>Algorith</u> :
- Klassisch : O(n^2)
- Divide - And - Conquer : O(nlg(n))
  - → Teile die Ebene durch eine Gerade parallel zur y-Achse, so daß jede Hälfte die gleiche Anzahl an Punkten beinhaltet.
  - → Suche rekursiv in jeder der beiden Hälften.
  - → Verschmelze die beiden Hälften und überprüfe ob es zwei Punkte gibt (in jeder Hälfte einen), die näher beieinander liegen, als das am nahesten beieinander liegende Paar in jeder Hälfte.

<u>Visualisierung</u> (Bild 7) :

3) **Heapsort**

Aufgabe: Sortiere eine Menge mit n Schlüsseln ihrer Ordnung nach.

Algorith:
- Baue Anfangsheap auf (binärer Baum), dabei muß die Heapeigenschaft erfüllt sein (Wert des Vaterknotens muß größer sein, als der seiner Kinder).
- Entferne Wurzel und mache neues Element zur Wurzel, und stelle die Heapeigenschaft wieder her.

Visualisierung (Bild 8):

4) K-Suchbaum

Aufgabe : Dient zur Speicherung mehrdimensionaler Objekte.
(z.B. Punkte in der Ebene - Kartesische Koordinaten)

Algorith : (Für K = 2)

- Gegeben : Punkte in der X/Y - Ebene.
- Wähle einen Punkt aus, und lege durch diesen eine Gerade parallel zur y-Achse.
  → 2 Ebenen
- Wähle einen weiteren Punkt und lege durch diesen eine horizontale Gerade, so daß diese die entsprechende Halbebene weiter unterteilt.
- Allgemein : ein Punkt, der in einer Region liegt, die durch eine horizontale Teilung entstanden ist, teilt diese Region vertikal und umgekehrt.

→ Teilungprozeß induziert einen binären Baum.

Visualisierung (Bild 9) :

5) **Balancierte Bäume** **((2,4)-Bäume)**

Aufgabe : Finden, Einfügen, und Löschen von Schlüsseln in möglichst kurzer Zeit.

Algorith :
- Jeder (innere) Knoten hat höchstens 4 und mindestens 2 Kinder ((2,4)-Eigenschaft)
- Einfügen eines Knotens kann die (2,4)-Eigenschaft verletzen : → Splitting
- Löschen eines Knotens kann die (2,4)-Eigenschaft verletzen : → Verschmelzen

Visualisierung (Bild 10) :

IV. Fazit

- Verwendung von 3D-Algorithmenanimationen :
  → **Ästhetik**
  → **Notwendigkeit**
  → **Verständnis durch weitere Information** : neue Einsichten
  (Bestätigung des Zippereffektes bei Shakersort – 1985)

- Einfach zu handhabende Tools sind notwendig (Polka-3D) um den Programmierer von der Komplexität aus den Bereichen 3D-Grafikkonzepte,Terminologie und Techniken abzuschotten.

- 3D-Algorithmenanimation ist ein sehr „junges“ Gebiet und größtenteils unerforscht.