Podpora poznámok PG3 21. 10. 2020, animácia, príklady (CASA, Szirmay-Kalos, Foley et al., Ruzicky, X3D)

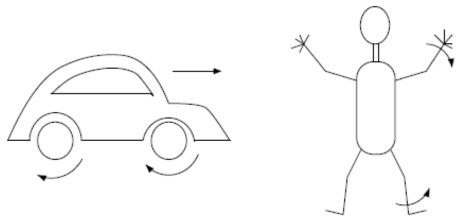


Figure 13.11: Examples of multi-segment objects

Recall that the motion of an individual object is defined by a time-varying modeling transformation matrix which places the object in the common world coordinate system. If the relative motion of object i must be defined with respect to object j, then the relative modeling transformation $\mathbf{T_{ij}}$ of object i must place it in the local coordinate system of object j. Since object j is fixed in its own modeling coordinate system, $\mathbf{T_{ij}}$ will determine the relative position and orientation of object i with respect to object j. A point $\vec{r}_i$ in object i's coordinate system will be transformed to point:

$$[\vec{r}_j, 1] = [\vec{r}_i, 1] \cdot \mathbf{T_{ij}} \quad \Longrightarrow \quad \vec{r}_j = \vec{r}_i \cdot \mathbf{A_{ij}} + \vec{p}_{ij} \qquad (13.57)$$

Transformation $\mathbf{T_{ij}}$ places object i in the local modeling coordinate system of object j. Thus, the world coordinate points of object i can be generated if another transformation — object j's modeling transformation $\mathbf{T_j}$ which maps the local modeling space of object j onto world space — is applied:

$$[\vec{r}_w, 1] = [\vec{r}_j, 1] \cdot \mathbf{T_j} = [\vec{r}_i, 1] \cdot \mathbf{T_{ij}} \cdot \mathbf{T_j}. \qquad (13.58)$$

In this way, whenever object j is moved, object i will follow it with a given relative orientation and position since object j's local modeling transformation will affect object i as well. Therefore, object j is usually called the **parent segment** of object i and object i is called the **child segment** of object j. A child segment can also be a parent of other segments. In a simulated human body, for instance, the upper arm is the child of the trunk, in turn is the parent of the lower arm (figure 13.12). The lower arm has a child, the hand, which is in turn the parent of the fingers. The parent-child relationships form a *hierarchy of segments* which is responsible for determining the types of motion the assembly structure can accomplish. This hierarchy usually corresponds to a *tree-structure* where a child has only one parent, as in the examples of the human body or the car. The motion of an assembly having a tree-like hierarchy can be controlled by defining the modeling transformation of the complete structure and the relative modeling transformation for every single parent-child pair (joints in the assembly). In order

Príklady (LS: tree-structure, Ru: sieť štruktúr, VRML/X3D scene graph: animation = modification), acyklický orientovaný graf (DAG)

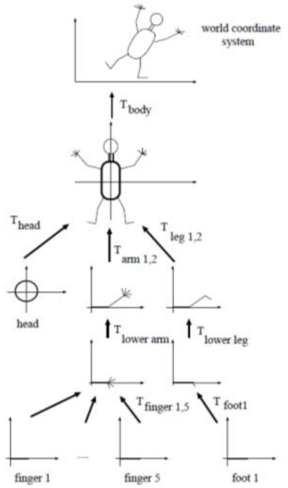
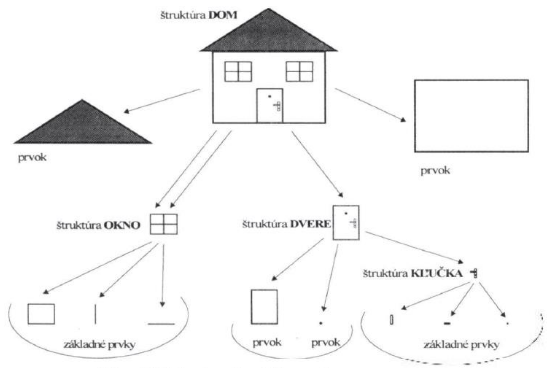
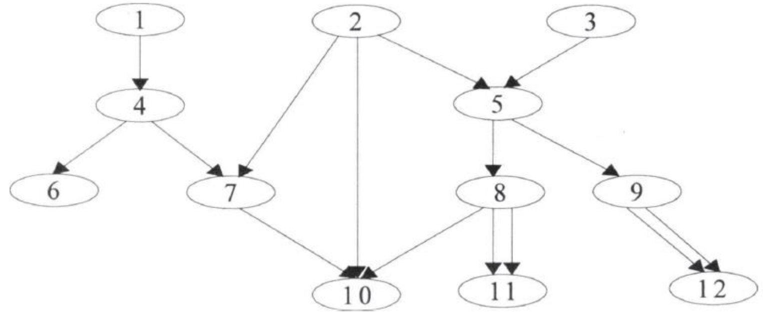


Figure 13.12: Transformation tree of the human body [LS]



Obr. 19.2 Viacúrovňová hierarchia



Obr. 19.3 Príklad siete štruktúr

[Ru, kap 19]

1. **Vybrané príklady** (10 minútové, YouTube v programe, napr. robot na s. 2, článok 6 *A Robot for Interactive Glove Puppetry Performance*)

HOT NEWS 13 ~ 15 October 2020 @ CASA the oldest international conference in computer animation and social agents in the world. It was founded in Geneva in 1988
http://casa2020.bournemouth.ac.uk/wp-content/uploads/2020/10/CASA2020-Programme-1.pdf


CASA 2020 Programme

October 13th ~ 15th 2020, Bournemouth, UK

Day 1 – October 13th, 2020 (all time in GMT)

12:00 – 12:15 Opening Ceremony

12:15 – 13:15 Keynote by Professor Taku Komura, Edinburgh University, UK

Tea Break 13:15 – 13:30

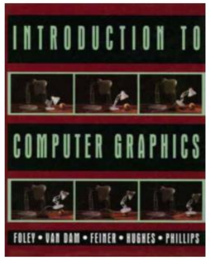
13:30 – 14:30 Keynote by Dr Changxi Zheng, Columbian University, US

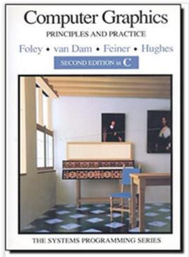
Day 2 – October 14th, 2020 (all time in GMT)

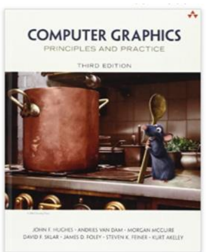
Track 1: Virtual Human Animation

Track 2: Virtual Reality (Part One)

<<< kliknite sem na druhé č. 6, Virtual Reality, napr. o „asymetrickej interakcii“

OLD NEWS, vydania „našej biblie“ 





2. **Vybraný príklad na interpoláciu**, obrázok Lasetter. Uvažujme dve vrchné polohy lopty a medzi nimi jednu spodnú. Lineárna interpolácia vypočíta **„chybné V“.**

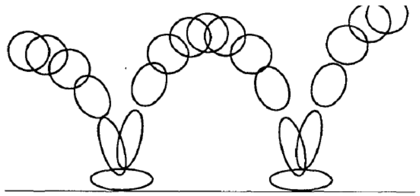


FIGURE 2. Squash & stretch in bouncing ball.

Ponaučenie z príkladu (originál vo Foley et al.): TREBA pohyb po krivke v 2D alebo na interpoláciu orientácií sférickú interpoláciu a kvaterniony v 3D. *Shoemake, Ken. "Animating Rotation with Quaternion Curves" (PDF). SIGGRAPH 1985.*

3. Vybraný príklad na artikulovanú štruktúru, skok po krivke z 2. vydania „biblie“ a chybný pohyb 2D lopty.

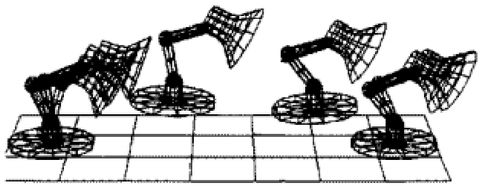


Fig. 20.25 Luxo Jr. is asked to jump from one position on the table to another. An initial path is specified in which Luxo moves above the table. Iterations of a variational technique lead Luxo to find a crouch–stretch–followthrough approach to the motion

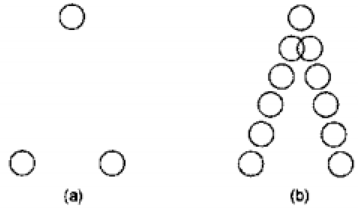
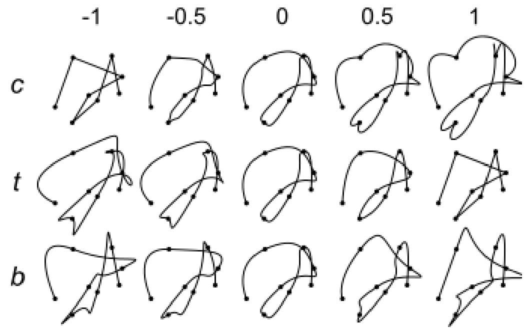


Fig. 21.1 Linear interpolation of the motion of a ball generates unrealistic results. (a) Three key-frame positions for the ball. (b) The resulting interpolated positions.

Interpolating the orientation of the rigid body is more difficult. In fact, even specifying the orientation is not easy. If we specify orientations by amounts of rotation about the three principal axes (called *Euler angles*), then the order of specification is important. For example, if a book with its spine facing left is rotated 90° about the x axis and then −90° about the y axis, its spine will face you, whereas if the rotations are done in the opposite order, its spine will face down. A subtle consequence of this is that interpolating Euler angles leads to unnatural interpolations of rotations: A rotation of 90° about the z axis and then 90° about the y axis has the effect of a 120° rotation about the axis (1, 1, 1). But rotating 30° about the z axis and 30° about the y axis does not give a rotation of 40° about the axis (1, 1, 1)—it gives approximately a 42° rotation about the axis (1, 0.3, 1)!

The set of all possible rotations fits naturally into a coherent algebraic structure, the *quaternions* [HAMI53]. The rotations are exactly the *unit quaternions*, which are symbols of the form $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where a, b, c, and d are real numbers satisfying $a^2 + b^2 + c^2 + d^2 = 1$; quaternions are multiplied using the distributive law and the rules $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$, $\mathbf{ij} = \mathbf{k} = -\mathbf{ji}$, $\mathbf{jk} = \mathbf{i} = -\mathbf{kj}$, and $\mathbf{ki} = \mathbf{j} = -\mathbf{ik}$. Rotation by angle ϕ about the unit vector $[b \quad c \quad d]^t$ corresponds to the quaternion $\cos\phi/2 + b\sin\phi/2\,\mathbf{i} + c\sin\phi/2\,\mathbf{j} + d\sin\phi/2\,\mathbf{k}$. Under this correspondence, performing successive rotations corresponds to multiplying quaternions. The inverse correspondence is described in Exercise 21.7.

Since unit quaternions satisfy the condition $a^2 + b^2 + c^2 + d^2 = 1$, they can be thought of as points on the unit sphere in 4D. To interpolate between two quaternions, we simply follow the shortest path between them on this sphere (a *great arc*). This spherical linear interpolation (called *slerp*) is a natural generalization of linear interpolation. Shoemake [SHOE85] proposed the use of quaternions for interpolation in graphics, and developed generalizations of spline interpolants for quaternions.



4. Vybraný príklad krivky, KB s parametrami t, b, c

Kochanek–Bartels spline

From Wikipedia, the free encyclopedia

In mathematics, a **Kochanek–Bartels spline** or **Kochanek–Bartels curve** is a cubic Hermite spline with tension, bias, and continuity parameters defined to change the behavior of the tangents.

Given $n + 1$ knots,

$\mathbf{p}_0, ..., \mathbf{p}_n,$

to be interpolated with n cubic Hermite curve segments, for each curve we have a starting point $\mathbf{p}_i$ and an ending point $\mathbf{p}_{i+1}$ with starting tangent $\mathbf{d}_i$ and ending tangent $\mathbf{d}_{i+1}$ defined by

$$\mathbf{d}_i = \frac{(1-t)(1+b)(1+c)}{2}(\mathbf{p}_i - \mathbf{p}_{i-1}) + \frac{(1-t)(1-b)(1-c)}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

$$\mathbf{d}_{i+1} = \frac{(1-t)(1+b)(1-c)}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{(1-t)(1-b)(1+c)}{2}(\mathbf{p}_{i+2} - \mathbf{p}_{i+1})$$

where...

t *tension*	Changes the **length** of the *tangent vector*
b *bias*	Primarily changes the **direction** of the *tangent vector*
c *continuity*	Changes the **sharpness** in change between tangents

Setting each parameter to zero would give a Catmull–Rom spline.

The source code found here of Steve Noskowicz in 1996 actually describes the impact that each of these values has on the drawn curve:

Tension	T = +1→ Tight	T = −1→ Round
Bias	B = +1→ Post Shoot	B = −1→ Pre shoot
Continuity	C = +1→ Inverted corners	C = −1→ Box corners

5.Vybraný príklad motivácie, prečo kvaterniony [Szirmay-Kalos] a schema MoCap

motion. In order to demonstrate this problem, suppose that an object located in [1,0,0] has to be rotated around vector [1,1,1] by 240 degrees and the motion is defined by three knot points representing rotation by 0, 120 and 240 degrees respectively (figure 13.6). Rotation by 120 degrees moves the x axis to the z axis and rotation by 240 degrees transforms the x axis to y axis. These transformations, however, are realized by 90 degree rotations around the y axis then around the x axis if the roll-pitch-yaw representation is used. Thus the interpolation in roll-pitch-yaw angles forces the object to rotate first around the y axis by 90 degrees then around the x axis instead of rotating continuously around [1,1,1]. This obviously results in uneven and unrealistic motion even if this effect is decreased by a C^2 interpolation.

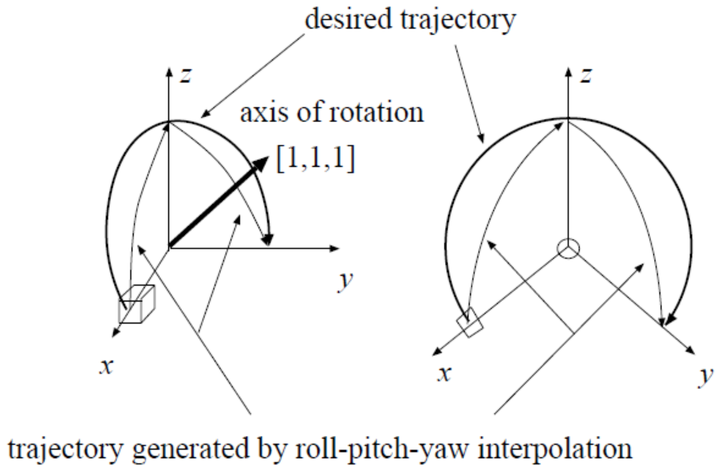


Figure 13.6: Problems of interpolation in roll-pitch-yaw angles

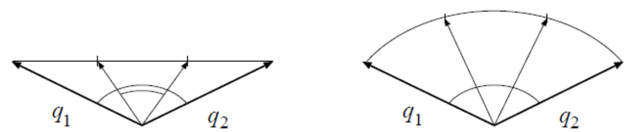


Figure 13.8: Linear versus spherical interpolation of orientations

Konkrétny príklad je na s. 47n v knihe Vince: Geometric Algebra.

as unit-size four-vectors which correspond to a 4D unit-radius sphere. An appropriate interpolation method must generate the great arc between q_1 and q_2, and as can easily be shown, this great arc has the following form:

$$q(t) = \frac{\sin(1-t)\theta}{\sin\theta} \cdot q_1 + \frac{\sin t\theta}{\sin\theta} \cdot q_2, \qquad (13.47)$$

where $\cos\theta = \langle q_1, q_2 \rangle$ (figure 13.9).

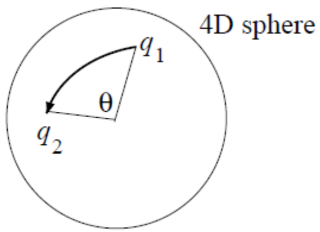


Figure 13.9: Interpolation of unit quaternions on a 4D unit sphere

54 3 Computer Animation

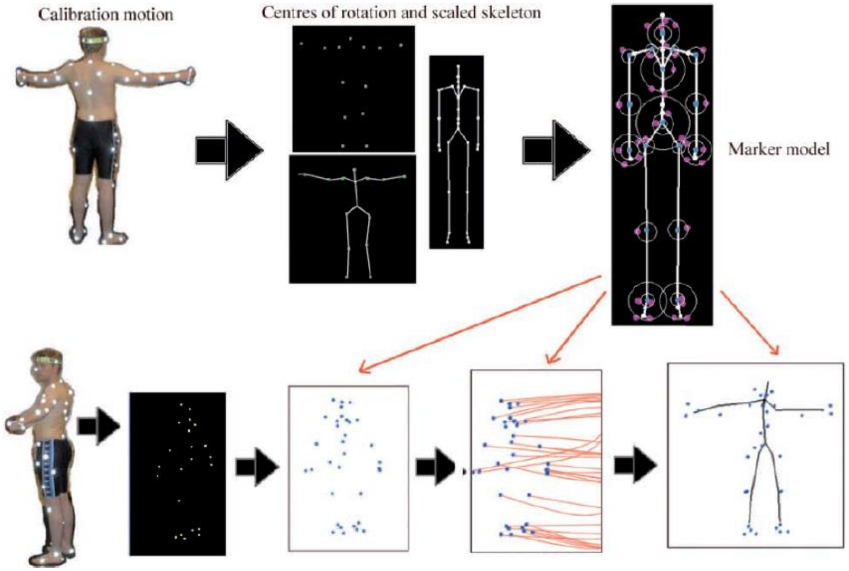
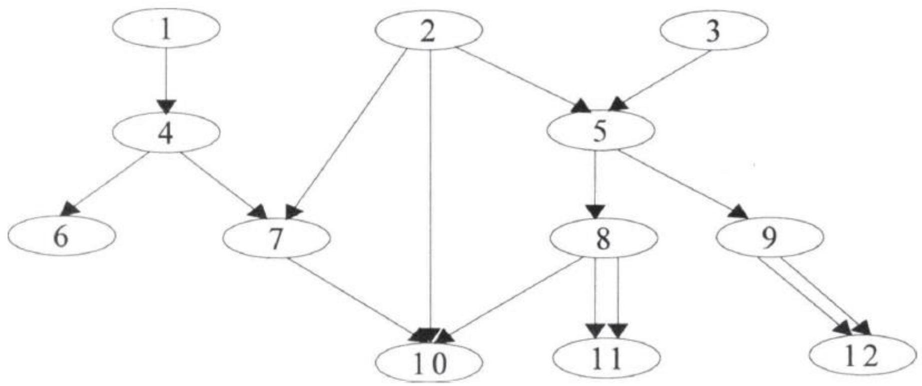


Fig. 3.2: Principles of optical motion capture

SLERP **[Szirmay-Kalos], MoCap** Gutiérrez - Vexo - Thalmann *Stepping into Virtual Reality*. Springer 2008.



Obr. 19.3 Príklad siete štruktúr

6.Vybraný príklad na skladanie transformácií a prechod sieťou štruktúr (scene graph traversal) [Ružický]

19.2.1 Modelujúce transformácie a orezávanie

Obrázok možno skladať z oddelených častí, z ktorých každú môžeme definovať v rámci jej vlastného **modelujúceho súradnicového systému** (*modelling coordinate system*, MC). Vzájomnú polohu oddelených častí získame pomocou jediných **svetových súradníc** (*world coordinates, WC*), na ktoré všetky MC zobrazíme **zloženou modelujúcou transformáciou** (*composite modelling transformation*). Priestor svetových súradníc sa chápe ako na stanici nezávislý abstraktný priestor. Zložená modelujúca transformácia C sa skladá z lokálnej modelujúcej transformácie L (definovanej v aktuálnej štruktúre) a globálnej modelujúcej transformácie G (čo je zložená modelujúca transformácia na začiatku prechodu štruktúrou zdedená z rodičovskej štruktúry). V nasledujúcom výklade použijeme dve funkcie, ktoré podrobne vysvetlíme neskôr: POST odošle štruktúru na zobrazenie na stanici a EXECUTE "vloží" štruktúru do otvorenej štruktúry. Štruktúra na najvyššej úrovni odoslanej siete štruktúr má na začiatku (každého prechodu sieťou štruktúr) G nastavenú na identickú transformáciu. L sa pre štruktúru takisto na začiatku (prechodu štruktúrou) nastaví na identickú transformáciu a modifikuje sa, keď sa pri prechode štruktúrou narazí na prvok SET LOCAL MODELING TRANSFORMATION 3 alebo SET LOCAL MODELING TRANSFORMATION. Zložená modelujúca transformácia sa vypočíta ako: C=G*L. Keď sa počas prechodu štruktúrou vyvolá podštruktúra, tak sa G aj L uschovajú a po návrate z prechodu podštruktúrou sa obe transformácie obnovia.

Prvky pre lokálne modelujúce transformácie vložíme do štruktúry funkciami: SET LOCAL TRANSFORMATION 3 (*matrix, type*) a SET LOCAL TRANSFORMATION (*matrix, type*), kde *matrix* je matica homogénnej transformácie (v prvom prípade 4*4, v druhom 3*3) a *type* určuje typ operácie s doteraz platnou lokálnou transformáciou. Označme LNEW ako výslednú lokálnu transformáciu, L pôvodnú lokálnu transformáciu a T transformáciu danú v *matrix*. Potom

LNEW = T pre *type* s hodnotou REPLACE

LNEW = L * T pre *type* s hodnotou PRECONCATENATE

LNEW = T * L pre *type* s hodnotou POSTCONCATENATE.

Vedľajší účinok tejto funkcie je samozrejme aj nastavenie zloženej modelujúcej transformácie na novú hodnotu CNEW = G * LNEW. Analogicky SET GLOBAL TRANSFORMATION 3 (*matrix*) a SET GLOBAL TRANSFORMATION (*matrix*) nastavia v stavovom zozname prechodu maticu globálnej transformácie na hodnoty podľa parametra *matrix*.

Príklad 19.1 Nastavenie transformácií pre štruktúry

```
/* Predpokladajme sieť štruktúr podľa obr. 19.3 a naviac, že štruktúry 4, 6, 7 obsahujú oi.
nasledujúce prvky pre nastavovanie transformácií a väzby na štruktúry v uvedenom
poradí: */

/* pre štruktúru 4 */
 SET LOCAL TRANSFORMATION 3 (L41, REPLACE)
 SET GLOBAL TRANSFORMATION 3 (G41)
 EXECUTE STRUCTURE (6)
 SET LOCAL TRANSFORMATION 3 (L42, POSTCONCATENATE)
 EXECUTE STRUCTURE (7)
/* pre štruktúru 6 */
 SET LOCAL TRANSFORMATION 3 (L61, REPLACE)
 SET GLOBAL TRANSFORMATION 3 (G61)
/* pre štruktúru 7 */
 SET LOCAL TRANSFORMATION 3 (L71, REPLACE)
 SET LOCAL TRANSFORMATION 3 (L72, PRECONCATENATE)
 EXECUTE STRUCTURE (10) /* prechod ďalšou podštruktúrou */
```

Aké budú hodnoty matíc zloženej, globálnej a lokálnej transformácie pre jednotlivé operácie pri prechode podsieťou štruktúry 4? Odpovede sú uvedené v nasledujúcej tabuľke, kde I znamená maticu identickej transformácie a operácie uvádzame len v skratkách.

Operácia	Lokálna transformácia	Globálna transformácia	Zložená transformácia	Poznámka
post(4)	I	I	I	(1)
local(L41,REP)	L41	I	L41	
global(G41)	L41	G41	G41*L41	
execute(6)	I	G41*L41	G41*L41	(2)
local(L61,REP)	L61	G41*L41	G41*L41*L61	
global(G61)	L61	G61	G61*L61	
návrat zo 6 do 4	L41	G41	G41*L41	(3)
local(L42,POST)	L42*L41	G41	G41*L42*L41	
execute(7)	I	G41*L42*L41	G41*L42*L41	(2)
local(L71,REP)	L71	G41*L42*L41	G41*L42*L41*L71	
local(L72,PRE)	L71*L72	G41*L42*L41	G41*L42*L41*L71*L72	
execute(10)	I	G41*L42*L41*L71*L72	G41*L42*L41*L71*L72	(2)
návrat z 10 do 7	L71*L72	G41*L42*L41	G41*L42*L41*L71*L72	(3)
návrat zo 7 do 4	L42*L41	G41	G41*L42*L41	(3)

Poznámky:

(1) Na začiatku prechodu sieťou štruktúr sa všetky modelujúce transformácie nastavia na identickú transformáciu. Treba si uvedomiť, že ak by sa aj v štruktúre 1 vyskytovali prvky na nastavenie modelujúcej transformácie, tieto by v tomto prípade nemali žiaden vplyv na globálnu a zloženú modelujúcu transformáciu, pretože prechod sieťou sa začína od štruktúry 4.

(2) Hodnoty matíc transformácie z predchádzajúceho riadku sa uložia, matica lokálnej transformácie sa nastaví na identickú, matica globálnej transformácie sa nastaví na hodnoty matice zloženej transformácie

(3) Pri návrate zo štruktúry sa obnovia uložené hodnoty.

7. Príklad postupu v deklaratívnom formáte. X3D is a royalty-free open standards file format and run-time architecture to represent and communicate 3D scenes and objects using XML. It is an ISO-ratified standard that provides a system for the storage, retrieval and playback of real time graphics content embedded in applications, all within an open architecture to support a wide array of domains and user scenarios. -- X3D: Extensible 3D Graphics for Web Authors by Don Brutzman, 2008. ... **Chapter Summary: Event Animation**

ROUTE connections and animation
Animation as scene-graph modification
Event-animation design pattern: 10-step process
Interpolation nodes

- TimeSensor and event timing
- ScalarInterpolator and ColorInterpolator
- OrientationInterpolator, PositionInterpolator, PositionInterpolator2D and NormalInterpolator
- CoordinateInterpolator, CoordinateInterpolator2D ...

https://www.web3d.org/example http://x3dgraphics.com/examples/X3dForWebAuthors/ https://www.web3d.org/x3d/what-x3d

Hello X3D Authors 10-step process

1. Pick target. The target node is a Transform, and the target field is *set_rotation*.

2. Name target. The Transform is named *DEF*='EarthCoordinateSystem'.

3. Check accessType and data type. As shown by the Transform node field-definition table in Chapter 3 and the X3D-Edit tooltip, the *set_rotation* field has type SFRotation.

4. Determine whether Sequencer or Script. These special node types are not applicable to this example, because the data type for *set_rotation* is SFRotation which is a floating-point type.

5. Determine which Interpolator. The animating OrientationInterpolator is named *DEF*="SpinThoseThings" and placed just before the Transform.

6. Triggering sensor. A triggering TouchSensor is added next to the geometry to be clicked, and then named *DEF*='ClickTriggerTouchSensor'.

7. TimeSensor clock. The TimeSensor is added at the beginning of the chain, named *DEF*='OrbitalTimeInterval' and has both the *cycleInterval* and *loop* fields set.

8. Connect trigger. Add ROUTE to connect the triggering TouchSensor node's *touchTime* output field to the clock node's startTime input field.

9. Connect clock. Add ROUTE to connect the clock node's *fraction_changed* output field to the interpolator node's set_fraction input field.

10. Connect animation output. Add ROUTE to connect the interpolator node's *value_changed* output field to the original target input field, *set_rotation*.

web|3D CONSORTIUM   X3D