TIEA311 Tietokonegrafiikan perusteet

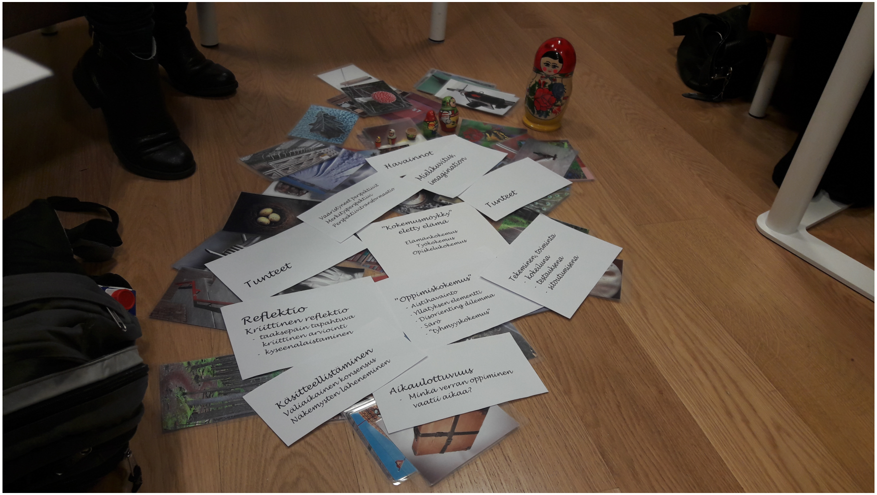
kevät 2019

("Principles of Computer Graphics" – Spring 2019)

TIEA311 Tietokonegrafiikan perusteet – kevät 2019 (“Principles of Computer Graphics” – Spring 2019)

Frontpage of the local course version, held during Spring 2019 at the Faculty of Information technology, University of Jyväskylä: `http://users.jyu.fi/~nieminen/tgp19/`

Greetings from YPE and APO (University Pedagogical Studies) – the resistance has gathered to contemplate revolution:



TIEA311 - Today in Jyväskylä

Plan for today:

- Usual warm-up and group discussion
- Try to address the most urgent issues
- Break – reset the brain.
- Then continue with the theory.

TIEA311 - Today in Jyväskylä

We start by discussion, reflection and **questions**!

Work in groups of 3 students if possible:

- Fast warm-up: 90 seconds evenly split between group members (30s each in groups of 3), no interruptions from others: Foremost feelings right now?
- Reflection: Silent work, solo, 1 minute, **list words on paper**: What have you learned during the last week? Or since the course started?
- Interaction: 1.5 minutes group discussion: Compare if you learned the same or different things? Do those things feel useful? Why or why not?
  $\rightarrow$ Sum it up classwide.
- Interaction: Group work, 1.5 minutes or less if talk ends: At the moment, what would be the most helpful thing to help you (or others!)?
  $\rightarrow$ Sum it up classwide, and try to address the findings.

TIEA311 - Today in Jyväskylä

What were the findings in group discussion?

What were found to be the most important issues to address right now?

$\rightarrow$ Classwide discussion is found on the lecture video.

NOTE: Even if you watch at home, please think about the same things and try to be in ”virtual dialogue” with those in classroom. Use pen and paper! I believe, more and more every day, that doing so will make your brain perform activities that help **your own learning**.

NOTE: Contemplate if you could watch the lecture videos with some friends who would also like to learn computer graphics? Get some pizza and coke if it helps you get to the mood(?).

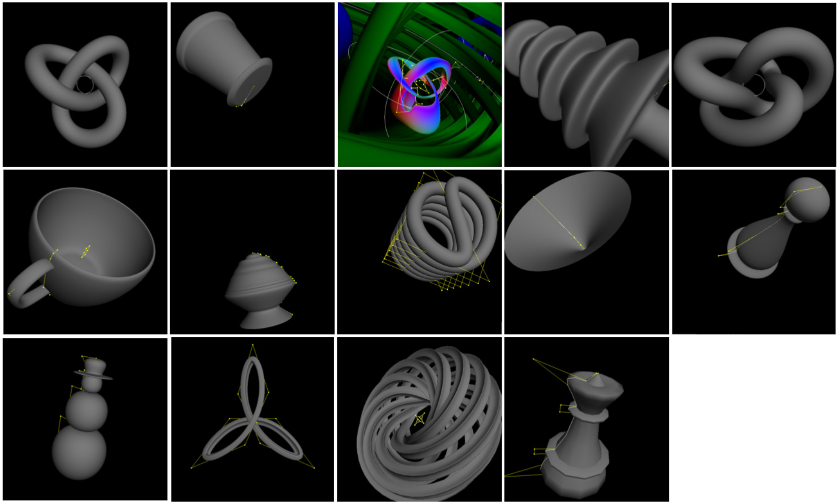
TIEA311 - Today in Jyväskylä

Plan for today:

- Usual warm-up and group discussion
- Try to address the most urgent issues
- Break – reset the brain.
- Then continue with the theory.

TIEA311

Towards Assignment 1, the first **real one**! Exercise answers from last year's students scoring 4/5 points or more:



TIEA311

Assignment 1 is about:

- What is a curve
- What is a surface
- How to model curved surfaces
- How to tessellate smooth shapes into a triangle mesh
- How to manage normals and orientations
- All this applicable to real-time animations, procedural content generation, and understanding the inner workings of modeling software we'd like to create for graphical artists and engineers.

Hold your horses! We must **understand** some stuff first!

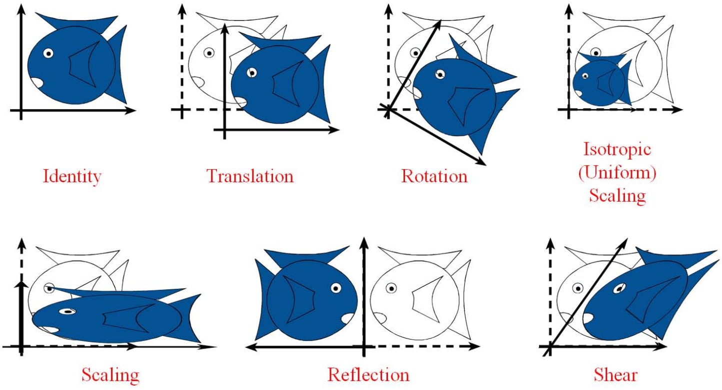
TIEA311 - Today in Jyväskylä

Brief recap of last time - to turn our minds towards the new.

$\rightarrow$ skim through previous slides **very quickly**.

Affine transformations

- Include all linear transformations
  - Applied to the vector basis
- Plus translation



Courtesy of Prof. Fredo Durand. Used with permission.

Matrix notation

- We know how to transform the vector basis

$$\begin{bmatrix} \mathcal{L}(\vec{b}_1) & \mathcal{L}(\vec{b}_2) & \mathcal{L}(\vec{b}_3) \end{bmatrix} = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$

- We will soon add translation by a vector $\vec{t}$

$$\tilde{p} \Rightarrow \tilde{p} + \vec{t}$$

Linear component

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

$$\Longrightarrow$$

$$\tilde{o} + \sum_i c_i \mathcal{L}(\vec{b_i}) = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & 0 \\ M_{21} & M_{22} & M_{23} & 0 \\ M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

- Note how we leave the fourth component alone

Translation component

$$\tilde{p} \Rightarrow \tilde{p} + \vec{t}$$

- Express translation vector t in the basis

$$\vec{t} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} \end{bmatrix} \begin{bmatrix} M_{14} \\ M_{24} \\ M_{34} \end{bmatrix}$$

Translation

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

$$\Longrightarrow$$

$$\tilde{o} + \vec{t} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & M_{14} \\ 0 & 1 & 0 & M_{24} \\ 0 & 0 & 1 & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

Full affine expression

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

$$\Longrightarrow$$

$$\tilde{o} + \vec{t} + \sum_i c_i \mathcal{L}(\vec{b_i}) = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

Which tells us both how to get a new frame ftM
or how to get the coordinates Mc after transformation

Questions?

More notation properties

- If the fourth coordinate is zero, we get a vector
- Subtracting two points:

$$\tilde{p} = \vec{f}^t \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} \qquad \tilde{p'} = \vec{f}^t \begin{bmatrix} c_1' \\ c_2' \\ c_3' \\ 1 \end{bmatrix}$$

- Gives us $\tilde{p} - \tilde{p'} = \vec{f}^t \begin{bmatrix} c_1 - c_1' \\ c_2 - c_2' \\ c_3 - c_3' \\ 0 \end{bmatrix}$

  a vector (last coordinate = 0)

More notation properties

- Adding a point

$$\tilde{p} = \vec{f}^t \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$ to a vector $$\vec{v} = \vec{f}^t \begin{bmatrix} c'_1 \\ c'_2 \\ c'_3 \\ 0 \end{bmatrix}$$

- Gives us

$$\tilde{p} + \vec{v} = \vec{f}^t \begin{bmatrix} c_1 + c'_1 \\ c_2 + c'_2 \\ c_3 + c'_3 \\ 1 \end{bmatrix}$$

a point (4th coordinate=1)

More notation properties

- vectors are not affected by the translation part

$$\begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 0 \end{bmatrix}$$

- because their 4th coordinate is 0
- If I rotate my moving car in the world, I want its motion to rotate
- If I translate it, motion should be unaffected

Questions?

Further Examples (Affine transforms)

Look at the implementation of the Matrix4f class in our example codes, found in the files `Matrix4f.cpp` and `Matrix4f.h`

Really, do it. (And keep doing it all the time!)

Observe how the fourth coordinate is used to implement 3D frames and affine transforms of points. And linear transforms of directions. A straightforward way to do many things is to build a proper 4x4 matrix and then multiply. **Not much code**, actually!

In Assignment 1 you will **avoid a lot of tears** by figuring out how (and when and why) to use the provided constructors and the operator $\star$ to transform points and frames suitably.

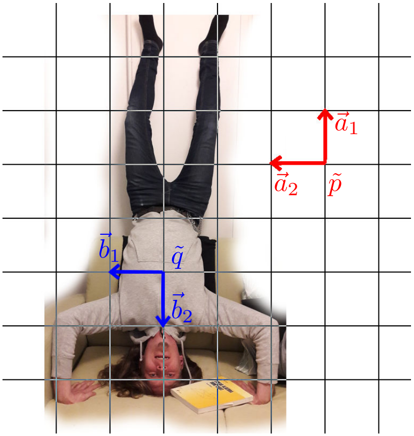
Also, this stuff **is a key thing** in **computer graphics**!

TIEA311

Your teacher **did** promise to stand on his head, if it could help you to learn graphics?

In order to avoid injury, the situation has been staged beforehand.

TIEA311

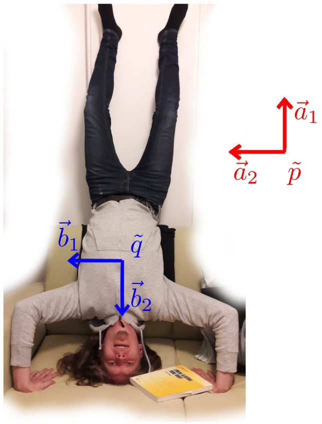


"Midterm!"

Using righthanded 2D coordinate system (y opens "left" of x) and visual inspection of red "world" frame $\vec{\mathbf{a}}$ and blue "model" frame $\vec{\mathbf{b}}$ fill in the matrix:

$$\vec{\mathbf{b}}^T = \vec{\mathbf{a}}^T \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

TIEA311



Observe:

- Affine frames transform **fine without coordinates**!
- Just scales and sums **relative to parent**.
- Only **points** and **directions** (origin and linear basis).

Distances?

- Need a concept of a **metric.** “Measure tape” in our usual, Euclidean 3D world. $\rightarrow$ live example on lecture video

TIEA311

Point of view ("independently discovered" by your teacher while thinking about this course and its primary target audience):

- Mathematics is structured much like careful object oriented design (and vice versa)! (Opinions may vary!)
- Math defines hierarchies of general and specific "classes" that model minimal, essential properties and methods found in worlds – real and imagined.
- **Vector space** models "arrows", **affine space** also "points". **Metric spaces** model distances but **not necessarily arrows**, … like **"interfaces"** of Java classes! (Can be **combined** or "multiply inherited" in a concrete class).
- Euclidean space $\mathbb{R}^3$ models distances and angles specifically in a Cartesian (orthonormal) coordinate system through dot product (which we'll learn soon-ish).
- Analogy from Assignment 0: C++ STL models a generic "stream" and specific "input stream" or "output stream" and finally an input stream instance called "cin" for you to "cin $>>$ x $>>$ y $>>$ z;". This **behaviour** works for **any** istream!

TIEA311

Point of view cont'd ("independently discovered" by your teacher while thinking about this course and its primary target audience):

- Mathematics is nothing to be afraid of. Instead, we should embrace it. And, for example, in graphics, we must.
- Mathematics has been incrementally developed, peer reviewed, and **thoroughly tested** for the last 2500 years or so. (Compare with the 70 years we've had computer programming.)
- Resultingly, for many purposes, mathematics Just Works.
- We can **apply math like we apply any class library**: by learning how to read its documentation and constructing suitable instances for our current tasks!

That said, let us open the "API of mathematics", on page "vector spaces, linear maps ($\approx$matrices), and their applications in cool CGI".

TIEA311 - Today in Jyväskylä

The time allotted for this lecture is now over.

Now: Break until tomorrow morning. Sleep if you have time.

But also **try to wake up and come to the lecture**!

We will pick up our thoughts soon enough!