Contact Information

• Raimond Tunnel – jee7@ut.ee

• Konstantin Tretjakov – kt@ut.ee
Organizational Information

- 15 seminars:
  - 3 introductory lectures
    What about 24th of February?
  - 9-11 student presentations
  - 2 project demonstrations
We hope that...

- ~15 seminars
  Attendance: 22.5h = 0.85 credits

- 1 seminar
  Preparation: 16h = 0.6 credits
  Conducting: 1.5h = 0.05 credits

- Project
  Everything: 40h = 1.5 credits
... but it may happen that ...

- ~15 seminars
  Attendance: 22.5h = 0.85 credits

- 1 seminar
  Preparation: 56h = 2.1 credits
  Conducting: 1.5h = 0.05 credits

- Project
  Everything: 0h = 0 credits

I read 3 books and am now a master of the subject.

Ain't nobody got time for that...
What am I even doing here?
What do I see?
What about this one?
This should be easy...
Regular seminars

- Listen to your fellow student's awesome presentation
- Ask questions, discuss
- $X > 1$ heads are better than one
Your seminar

- Choose an interesting CG topic
- Make the seminar fun and interactive
- Present some applications / demos
- Workshop
Sidetrack: Gamma correction
Sidetrack: Gamma correction

Sidetrack: Bloom effect

Need for Speed: Most Wanted

Elephant's Dream

Hitman: Absolution

Warframe: https://www.youtube.com/watch?v=gYHxhlvEyHk
Back to the main track
How do I choose a topic?

- I just gave you two possibilities:
  - Shader effects (like the Bloom effect)
  - Gamma correction
- Read something and find interesting topics
  - OpenGL's Red Book
  - GPU Gems
  - More "sophisticated" literature
- Continue on some already discovered theme
  - My example: Procedural tree generation?
How do I choose a topic?

- Continue on some already discovered theme
How to choose a topic?

- OpenGL ver 3.0 & 3.1
- Practical
- Basic topics:
  - Viewing
  - Color
  - Lighting
  - Blending
  - Textures
  - Buffers
How to choose a topic?

- Advanced topics:
  - Display lists (perf.)
  - Tessellation
  - Quadrics
  - Evaluators (curves & surfaces)
  - NURBS
How to choose a topic?

• OpenGL ver 4.3
  • Lots of new techniques and topics.
  • For example:
    – Tessellation shaders
    – Geometry shaders (access to all vertices)
    – Procedural texturing

How do I choose a topic?

- Covers all topics already mentioned and more
- Math heavy, but most of it you should be at home with
## Extra conditions!

<table>
<thead>
<tr>
<th>First time participant</th>
<th>Returning participant</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSc, MSc</td>
<td>MSc, PhD</td>
</tr>
<tr>
<td>No additional requirements – you can choose any CG-related topic.</td>
<td>Your topic should be related to several scientific articles / a book.</td>
</tr>
</tbody>
</table>

ACM SIGGRAPH (Special Interest Group on GRAPHics and Interactive Techniques):
http://www.siggraph.org/
Previously...

PREVIOUSLY, ON SCRUBS...

Oh Wait, This Ain't Scrubs...
Procedural Geometry

- 2D Sprite Geometrization
  1) Boundary detection

![Boundary detection process](image)

2) Now we have a polygon with many vertices.

10285 pixels vs 3401 pixels
Procedural Geometry

• 2D Sprite Generation
  • Convex hull algorithm

You can run this to some extent in order to get rid of vertices.

If you complete the algorithm, you will still have many transparent pixels.

• Triangulation

Y-monotone polygon – every horizontal line intersects polygon at most twice.

Partition the convex hull polygon into y-monotone polygons.

\[ O(n \cdot \log(n)) \]
Procedural Geometry

• Procedural Landscapes
  • Heightmap
  • Vector field
  • Voxel based
    – Marching cubes
    – Marching tetrahedra
    – Surface nets

To make voxels into polygons
Procedural Geometry

- Flora
  - L-systems
  - Superformula
- Destruction
  - Glass
  - Voronoi diagrams
    - Fortune's algorithm
Procedural Geometry

- Procedural Cities
Shaders & Deferred Shading

- Shader is a computer program that is used to do shading (rendering effects, color etc).

![Diagram]

- Describe vertices (position, texture, color)
- Describe pixels (color, depth)
Shaders & Deferred Shading

- Geometry Shader – modifies geometry
- Tessellation Shader – splits triangles
Shaders & Deferred Shading

• Forward rendering (default):

• Deferred rendering
  • 1st pass – collect relevant data for pixels
Shaders & Deferred Shading

- Deferred shading
  - 2nd pass – use the collected data to shade pixels
Shaders & Deferred Shading

- Deferred lighting – accumulate light intensity

Each point light area is bounded by a sphere (approximate)
Color Blending

- **RGB**
  - RGB

- **HSV and HSL**

- **Alpha channel**
  - Opaqueness of a pixel
  - Used for blending of layers of images
  - Multiple blending modes
  - Pre-multiplied and post-multiplied alpha.
Color Blending

- Photoshop blending modes:
  - Normal \( f(a, b) = b \)
    
     Combined with some alpha compositing.
  - Dissolve – random transparency for pixels
  - Multiply \( f(a, b) = ab \)
  - Screen \( f(a, b) = 1 - (1 - a)(1 - b) \)
  - Overlay
    \[
    f(a, b) = \begin{cases} 
    2ab, & \text{if } a < 0.5 \\
    1 - 2(1 - a)(1 - b), & \text{otherwise} 
    \end{cases}
    \]
Color Blending

- Post-multiplied alpha – convex combination of colors based on alpha.

\[
\begin{align*}
\text{out}_A &= 1 \\
\text{out}_{RGB} &= \text{src}_{RGB}\text{src}_A + \text{dst}_{RGB}(1 - \text{src}_A)
\end{align*}
\]

When background \(\text{src}\) is opaque.

- Pre-multiplied alpha – multiply color values with alpha when image is stored.

\[
\begin{align*}
\text{out}_A &= \text{src}_A + \text{dst}_A(1 - \text{src}_A) \\
\text{out}_{RGB} &= \text{src}_{RGB} + \text{dst}_{RGB}(1 - \text{src}_A)
\end{align*}
\]

When background \(\text{src}\) is opaque.
Color Blending

- Pre-multiplied alpha is useful for:
  - Scaling images

<table>
<thead>
<tr>
<th></th>
<th>Known value</th>
<th>Interpolated value</th>
<th>Known value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post</td>
<td>(1.0, 0.0, 0.0, 1.0)</td>
<td>(1.0, 0.5, 0.5, 0.5)</td>
<td>(1.0, 1.0, 1.0, 0.0)</td>
</tr>
<tr>
<td>Pre</td>
<td>(1.0, 0.0, 0.0, 1.0)</td>
<td>(0.5, 0.0, 0.0, 0.5)</td>
<td>(0.0, 0.0, 0.0, 0.0)</td>
</tr>
</tbody>
</table>

- Associativity for blending operations

\[ A \ast B \ast C \ast D = (A \ast B) \ast (C \ast D) = A \ast (B \ast C) \ast D \]
Color Blending

- Sander's project:
Post-Processing Effects

- Apply an effect after the scene has been rendered

- Color grading
- Antialiasing
- Bloom
- Motion Blur
- Lens Flare
- Depth of Field
- Bokeh
- ...

[Images of visual effects illustrations]
Anti-Aliasing

• Aliasing

• Solutions
  • Supersampling
  • Multisampling (MSAA)

Render at higher resolution, downscale later.

Do many samples per pixel, but execute fragment / pixel shader at most once.
Anti-Aliasing

- **Solutions**
  - **Post-processing techniques**
    - FXAA – **Fast Approximate Anti-Aliasing**
    - SMAA – **Subpixel Morphological Anti-Aliasing**
  
  - Faster than MSAA
  - Can be used with deferred shading.
Non-Photorealistic Rendering

- Shading with strokes
- Automatic painting
- Cartoon shading
- Gooch shading
- Cel shading
3D Scan Data

Pointcloud in MeshLab

Constructed model in Blender

http://ikuz.eu/2014/04/03/proof-of-concept-from-3d-scanner-to-animated-model/
3D Printing
Procedural Maps, Terrains and Environments

- Storage space
- Replay value
- *Imperfect factory*

- Problems
  - Homogeneous
  - Impassable areas
  - Will it look good?
Procedural Maps, Terrains and Environments

- Symmetric map for RTS
- Noise height generation followed by:
  - Voronoi
  - Arbitrary splines
  - Nonlinear height
  - ...
- Foliage based on steepness
- Erosion and sand dune formation
- Prevalence of rivers based on latitude
Data Visualization

- Interesting data visualizations:
  - Newcomb fraternity graph
  - 3D plots
  - Technical visualization
Data Visualization

- Human perception

<table>
<thead>
<tr>
<th>Quantitative</th>
<th>Ordered</th>
<th>Categorical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Position</td>
<td>Position</td>
</tr>
<tr>
<td>Length</td>
<td>Lightness</td>
<td>Hue</td>
</tr>
<tr>
<td>Angle</td>
<td>Saturation</td>
<td>Texture</td>
</tr>
<tr>
<td>Slope</td>
<td>Hue</td>
<td>Connection</td>
</tr>
<tr>
<td>Area</td>
<td>Textures</td>
<td>Containment</td>
</tr>
<tr>
<td>Volume</td>
<td>Connection</td>
<td>Lightness</td>
</tr>
<tr>
<td>Lightness</td>
<td>Containment</td>
<td>Saturation</td>
</tr>
<tr>
<td>Saturation</td>
<td>Length</td>
<td>Shape</td>
</tr>
<tr>
<td>Hue</td>
<td>Angle</td>
<td>Length</td>
</tr>
<tr>
<td>Texture</td>
<td>Slope</td>
<td>Angle</td>
</tr>
<tr>
<td>Connection</td>
<td>Area</td>
<td>Slope</td>
</tr>
<tr>
<td>Containment</td>
<td>Volume</td>
<td>Area</td>
</tr>
<tr>
<td>Shape</td>
<td>Shape</td>
<td>Volume</td>
</tr>
</tbody>
</table>

- Color wheel: Red, Orange, Yellow, Purple, Blue, Green
Data Visualization

- Perspective projection distorts
- Avoid animation – hard to remember
- Noticing change requires attention
- Use sparklines to show change
Data Visualization

- CONREC algorithm for map contour
- ThemeRiver stacked graph
- Parallel coordinates
Soft Body Physics and Fractures

- Soft body vs:
  - Rigid body
  - Fluid
  - Particle system

- Soft body:
  - Deformable
  - Retains original shape
  - Computationally expensive
Soft Body Physics and Fractures

- Finite element simulation
- Shape matching
- Spring-mass model
  - Many particles with mass (repel)
  - Springs between particles (attract)

All have some constraints and a solver that iterates until constraints are satisfied.
Soft Body Physics and Fractures

• Fractures:
  • Premade fractured model
    – Pieces are glued together
  • Fracture mapping
    – Object is decomposed based on mapping

• Real-time fracture generation
  – Can account for the point of impact
Soft Body Physics and Fractures

• Different approaches:
  • Adaptive fracture refinement
    – Uses material strength field combined with stress field.
    – https://www.youtube.com/watch?v=WaKbMaBJa2Q
  • Adaptive fracture simulation of multi-layered thin plates
UX in Graphical Applications

• 5 competencies of UX design:
  • Information architecture
    – Interface structure and navigation scheme
  • Interaction design
    – Page level layout, user task fulfillment, component flow
  • Usability engineering
    – What users expect and what is there?
  • Visual design
    – Consistent visual treatment of elements
  • Prototype engineering
    – Composition of proposed interactive concepts
UX in Graphical Applications

XCOM: Enemy Unknown UI map
UX in Graphical Applications

- Wireframe and state chart.
Still confused?
You can...

- ... pick any topic from previous year
- ... pick some other CG related topic
World is a vast and mysterious place!

Ok, so I have a topic now...

- Look for materials
- Investigate, research
- Find examples
- Try it out yourself
- Present your findings
- Engage others
  - Discussion
  - Interactive demo
  - Workshop
What about the project?

- Interactive demo on the same topic as your seminar
What about the project?

- Advance something you've already done
What about the project?

• Can, of course, be a team / group effort!
What about the project?

- Do something fun and exciting

You can do it!
Want to do a LARGER project?

- Computer Graphics Project (MTAT.03.316)
  - 3 credits course
  - Consists entirely of a project
  - Work on your own idea throughout the semester
  - Roughly 6h per every 2 weeks
- https://courses.cs.ut.ee/2016/cg-project/spring
I didn't understand >70% of what you said...

• Don't worry about it!
• Pick a topic that suits your knowledge base
• Your topic may very well be:
  • Rasterization of triangles
  • Comparison of lighting models
  • How to do simple shadows?
  • Raytracing explained
  • etc
I don't even know where to start!

- There will be 2 more introductory lectures about the basics.

- Check out slides and tasks from Computer Graphics MTAT.03.015:
  https://courses.cs.ut.ee/2015/cg/spring
  https://courses.cs.ut.ee/2015/cg/fall

- Check out the slides from the previous seminar:
  https://courses.cs.ut.ee/2015/cg-sem/spring/Main/Seminars
  https://courses.cs.ut.ee/2015/cg-sem/fall/Main/Seminars

- Find some online tutorial that seems manageable for you and try it out.
Questions?
List of some topics

1. **Color blending** – What happens when there are transparent objects in your scene?
2. **Lighting models** – What are the common models? Where and when are they used?
3. **Texturing** – How can one sample from a texture? What kinds of artefacts may appear?
4. **Curves** – Why are they important in CG? What about curved surfaces?
5. **Global illumination** – Pick one or compare different methods: Radiosity, path tracing, photon mapping.
6. **Realtime realistic rendering** – Provide an overview of the common methods or pick some effect (light, wetness, fog, fur / hair) and find out how it's rendered realistically in real time.
7. **Non-realistic rendering** – Where is it used and how is it achieved? Realtime vs prerendered?
8. **Tessellation** – How can this be done in OpenGL 4?
9. **Post-processing effects** – What effects are there? When and how are they used?
10. **Procedural generation** – Where and how is it used? How to apply procedural textures to procedurally generated meshes?