Modern Technologies & Lighting

Jaanus Jaggo
Game engines have a lot of built-in technologies
Evolution of PC game graphics
Then vs Now

Not just the number of polygons

2004

2017
Importance of light

Simulated light behaviours:

- Shadows
- Reflections
- Ambient light
- Atmospheric scattering
- Physically correct materials

https://learnopengl.com/PBR/Theory
History - 2.5D

Looks like 3D but not made of polygons.

History - Early 3D games

Voodoo1 (3dfx)
1996
First pure 3D GPU

GeForce 256 (Nvidia)
1999

Quake 1996

GTA 3 - 2001

Textured materials with baked lights and every polygon was counted.
History - 3D with real-time lighting

Doom 3
2004
Stencil shadows

Elder’s Scrolls Oblivion
2006
History - Post effects

Battlefield 2
2005

Fallout 3
2008

Hitman Absolution
2012
History - Deferred rendering

- Cheaper lighting
- More post-effects

Crysis 2
2011
History - Deferred rendering

http://www.adriancourreges.com/blog/2016/09/09/doom-2016-graphics-study/

http://www.adriancourreges.com/blog/2015/11/02/gta-v-graphics-study/
Future - raytracing everything?
Light

Significant role in 3D games.

**RYSE: Son of Rome** scene with and without lighting:
Introduction to lighting in 3D games

Real world light is a complex thing

John Carmack explaining light in games: https://youtu.be/P6UKhR0T6cs
Rendering process

Window -> Analysis -> Frame Debugger
Light baking

Lightsources

1. Directional light:
   No falloff

2. Point light:
   Has falloff

3. Spot light:
   Has falloff, funnel shape

4. Ambient light:
Spot Light source effects

IES profiles for spot lights - provided by real world light manufacturers

Volumetric light beams

**Point Light** source effects

Sphere and tube (capsule) lights

Free alloy shaders:
https://assetstore.unity.com/packages/vfx/shaders/alloy-physical-shader-framework-11978
Lightsources

Area lights - real world light sources, have shape and size.
https://www.youtube.com/watch?v=ZLRgEN7AQgM

Article: https://eheitzresearch.wordpress.com/415-2/
Other light technologies

- Shading models
- Shadows
- Real time Global Illumination
- Reflections
Shading

Early games:
- Baked
- Simplistic light model (eg. Phong)
Shading

Physically-Based rendering (PBR) - more physically correct computations
Shading

**Metalness** - is the material metal or nonmetal
Shading

Energy conservation - object can not reflect more light than it receives.
Shading

**Fresnel** - the percentage of light that a surface reflects at grazing angles.
Shadows
Shadows

**Shadow mapping** - the scene is rendered from the perspective of the light source (shadow map).
Shadows

The quality depends on the resolution of the shadow map.
Shadows

**Cascaded Shadow Maps** -> multiple shadow maps with different resolutions. (More precision close to camera)
Advanced shadow sampling

Next gen soft shadows:
Global Illumination

Global Illumination (GI) - ambient light bouncing off the surfaces
Global Illumination

Raycasting

It can be done in real-time but it is very very slow and noisy.
Global Illumination

Bounce lighting can be baked into light probes (points in space). Dynamic objects will sample light from these points.

Brackeys video about light probes: https://youtu.be/_E0JXOZDTKA
Global Illumination

Unity -> Enlighten  Baked static geometry + dynamic light sources

Deprecated
Global Illumination

Unity -> Enlighten  P.A.M.E.L.A

Deprecated
Global Illumination

Both Unreal and Cryengine have real-time voxel cone-tracing GI solutions -> too slow for older computers

Unreal engine elemental demo https://www.youtube.com/watch?v=MOvfn1p92_8
Global Illumination

Voxel Cone Traced Global Illumination

Unity SEGI plugin: [https://github.com/sonicether/SEGI](https://github.com/sonicether/SEGI)
Reflections
Reflections

**Planar reflections** - camera renders scene below the surface

Performance expensive, only works with planar surfaces (not so good for wavy water).
Reflection probes - the surrounding is rendered into a cubemap.

Good results near the reflection probe.

More information about cubemaps.
Reflections

Parallax-corrected cubemaps:
https://www.youtube.com/watch?time_continue=72&v=Bvar6X0dUGs

Same cubemaps but projected differently.
Reflections

**Screen space reflections** - reflections sampled from already rendered scene.

Artifacts near the edges of the screen
Reflection disappears when object moves out of the screen or is occluded
Other technologies

- Geometry manipulation - Chamfered edges
- Volumetric effects - Fog
- Particles
- Post-effects
Limited texture size

UV layout starts simple and innocent

https://www.reddit.com/r/starcitizen/comments/3ogi3o/im_an_tech_artist_in_the_industry_and_id_love_to/
Limited texture size

**Problem:** texture space requirement grows depending on:
- Model scale
- Model complexity
- How close the player can get

https://www.reddit.com/r/starcitizen/comments/3ogi3o/im_an_tech_artist_in_the_industry_and_id_love_to/
Limited texture size

Ok, what about the Gladiator ship from Star Citizen

How large texture would it need?

https://www.reddit.com/r/starcitizen/comments/3ogi3o/im_an_tech_artist_in_the_industry_and_id_love_to/
Limited texture size

**Solution:** tiling textures + decals

[Image of tiling textures and decals]

https://www.reddit.com/r/starcitizen/comments/3ogi3o/im_an_tech_artist_in_the_industry_and_id_love_to/
https://www.reddit.com/r/starcitizen/comments/3ogi3o/im_an_tech_artist_in_the_industry_and_id_love_to/
Texture sampling

Sampling texture based on world coordinates (triplanar sampling):

- **Faster workflow** - Unwrapping and texture painting can be avoided

https://www.facebook.com/groups/IndieGameDevs/permalink/10154835397656573/
Fog

Historically used to hide the limited render distance

Adds sense of depth
fog

Fog particles
Fog

**Volumetric fog** - scatters light, creates light rays
Fog

**Volumetric fog** - scatters light, creates light rays

Unity implementation: [https://github.com/Unity-Technologies/VolumetricLighting](https://github.com/Unity-Technologies/VolumetricLighting)
Particles

Large number of tiny sprites or objects - simulate “fuzzy” phenomena
Particles

Modern particles:
● Emit light
● Collide with room geometry
● Distort background

GPU particles -> only available in Unreal Engine
Boids algorithm

Boids originally developed to simulate the flocking behavior of birds.
Fish simulation by Vetemaa: [https://vetemaa.github.io/fish-simulation/](https://vetemaa.github.io/fish-simulation/)

- **separation**: steer to avoid crowding local flockmates
- **alignment**: steer towards the average heading of local flockmates
- **cohesion**: steer to move towards the average position (center of mass) of local flockmates
Post effects

Postprocess effects - applied after the whole screen has been drawn.
Post effects

**Postprocess effects** - applied after the whole screen has been drawn.

**Deferred shading** - light is calculated based on previously rendered buffers

![Albedo color](image1)

![Surface normals](image2)

![Depth buffer](image3)

![Material roughness](image4)
Other post effects

**Depth of field** - blurs objects in the distance
Other post effects

**Ambient occlusion** - darkens corners and cavities
Other post effects

**Ambient occlusion** - darkens corners and cavities
Other post effects

**Bloom** - makes light pixels bleed out of their boundaries

Requires HDR

Use HDR colors as well
Other post effects

**Distortion** - distorts and curves the whole image

Distortion shader
Fur

GDC talk about fur rendering: https://youtu.be/bl61E2j_q-U
Some GC theses made in ATI

Diana Algma MSc

Siim Raudsepp BSc

Jaagup Kuhi BSc

Marko Täht MSc
How to study these technologies?

MTAT.03.015 Computer Graphics

MTAT.03.305 Computer Graphics Seminar
MTAT.03.328 Computer Graphics Project
Conclusion

This was just a small selection of modern 3D techniques.

Some sources that explain modern effects in video games:
http://www.adriancourreges.com/projects/
https://simonschreibt.de/game-art-tricks/
https://www.youtube.com/c/DanMoranGameDev/videos
Lecture task

- Find a picture about a cool video game effect.
- Write how similar thing can be achieved.
- Add a link to a source that describes its implementation in more details.

Example:

In Doom 3, the light source glow effect was implemented by using geometry with semi-transparent material. The geometry is moved based on camera locations such that the gradient always faces the player.

Source: [https://simonschreibt.de/gat/doom-3-volumetric-glow/](https://simonschreibt.de/gat/doom-3-volumetric-glow/)
Next lecture procedural generation