Simulations in video games

Henrik Holst

2014-10-09

"Rocket science is kind of simple, [...]
compared to video game development."

/John Carmack, Armadillo Aerospace & Id Software, 2008.

Outline

- Short about me and EA DICE.
- Three examples of simulations in video games:
    1. Audio raycast system (sound)
    2. Physically based sky (rendering)
    3. Screen space reflections (rendering)

I will only have time to cover two of these examples.

Henrik Holst

- PhD in numerical analysis, KTH 2011.
- Research subject *Multiscale methods for wave propagation*.
- Since Feb. 2014 working as software engineer at EA DICE.
- Previously, development engineer at COMSOL.

About DICE

- Founded 1992 in Sweden.
- Digital Illusions Creative Entertainment AB.
- Electronic Arts acquired DICE in 2004.
- Today over 500 employees.
- Recent AAA titles: *Battlefield* series and *Mirror's Edge*.
- Working with Lucas Film on *Star Wars: Battlefront*.

Previous titles

- 1992 Pinball Dreams (First released game)
- ...
- 2002 *Battlefield* 1942 (Refractor Engine)
- ...
- 2008 Battlefield: Bad Company (Frostbite engine)
- ...

Frostbite game engine

- Created by DICE
  - Developed by Frostbite DICE.
  - Catering many EA studios.
- Current version: *Frostbite 3*
  - FrostEd; Backend Services; and Runtime.
  - Created for *First person shooters* (FPS).
  - Customizable to other genres such as racing and real-time strategy.
- Artists creates or import *game assets* with FrostEd.
- Frostbite covers all aspects of development including:
  - Audio, Animation, Cinematic, Scripting,
  - Artificial intelligence, Physics, Destruction,
  - Rendering, Visual effects, . . .

Simulations in video games

"Simulation is the imitation of the operation of a real-world process or system over time."

/Discrete-Event System Simulation. ISBN 0-13-088702-1.

Simulations in video games

- Video game simulations should be realistic.
- *Not* always physically based.
- Some video game "simulations" are intentionally unrealistic:
  - Laser sounds in space.
  - The player has many lives.
  - Never needs to eat. Never gets tired.
  - ...

Audio raycast system

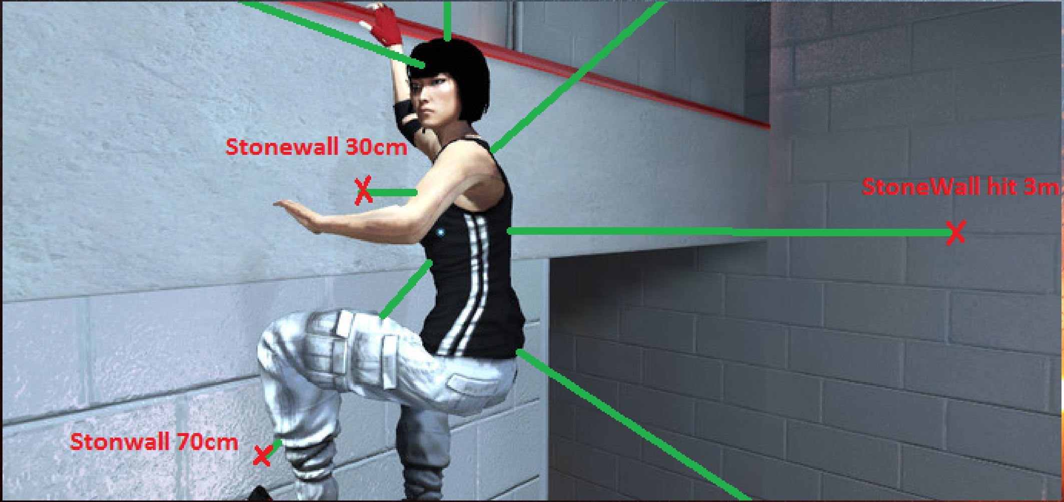


Image from Mirror's Edge 2008. Programmer art added.

Audio raycast system

- Physically correct simulation of acoustic sound wave propagation not yet possible on current hardware.
- Frostbite has a rich DSP functionality to control wave output.
- Ad hoc acoustic model in DSP; parametrized by:
  - Obstruction.
  - Sound position and relative velocity.
  - Listener/sound source combination: indoor, outdoor?
  - Material properties.
  - ...

Audio raycast system

- Sits on top of physics engine in Frostbite.
- Workhorse in ray cast algorithm: *collision detection*.
- Simplified *physics* mesh used.

Collision detection

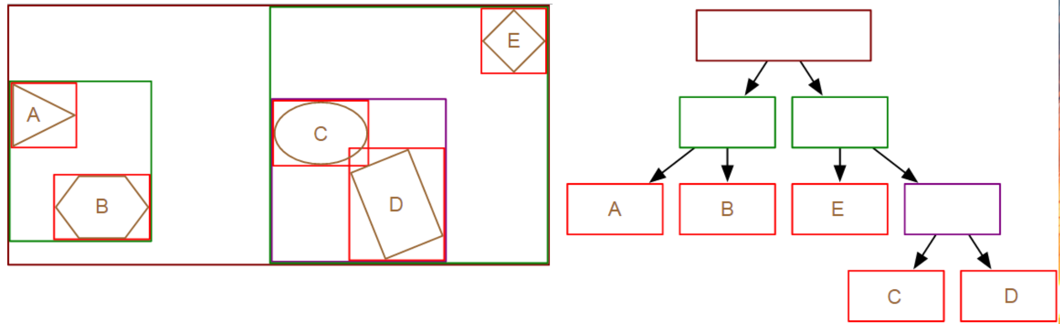
- Determine (pairwise) points of intersection between objects.
- Straight forward: N objects $\Rightarrow \frac{N(N-1)}{2}$ collision tests.
- Divide into two steps: *broad phase* and *narrow phase*.
  - Broad phase: Approximation. Efficiently eliminate pairs that *certainly* does not collide.
  - Fine phase: Exact geometric collision tests on remaining objects.

Broad phase

Example of broad phase data structure:

- Hierarchical tree of *axis aligned bounding boxes* (AABBs).
- Incrementally updated each simulation tick.
- Searching $\mathcal{O}(\log N)$ complexity.

AABB tree (borrowed from ncollide.org)



Fine phase

- Fine phase algorithm on remaining objects.
- Special algorithms for each kind of shape-to-shape collision test.
- Suitable for GPU implementation.
- Additional details on Line-plane intersection on Wikipedia.

Fine phase (cont.)

Example: Line to surface intersection.

A plane Π represented by 3 non-collinear points $\vec{p}_u, \vec{p}_v$ and $\vec{p}_0$:

$$\Pi = \left\{ \vec{p}_0 + u(\vec{p}_u - \vec{p}_0) + v(\vec{p}_v - \vec{p}_0) \mid u, v \in \mathbb{R} \right\}.$$

The line ℓ through the points $\vec{l}_1$ and $\vec{l}_0$, in parametric form:

$$\ell = \left\{ (1-t)\vec{l}_0 + t\vec{l}_1 \mid t \in \mathbb{R} \right\}.$$

Fine phase (cont.)

Finding the intersection between Π and ℓ means we need to solve for u, v and t:

$$u(\vec{p}_u - \vec{p}_0) + v(\vec{p}_v - \vec{p}_0) + t(\vec{l}_0 - \vec{l}_1) = \vec{p}_0 - \vec{l}_0,$$

corresponding to a 3-by-3 linear system,

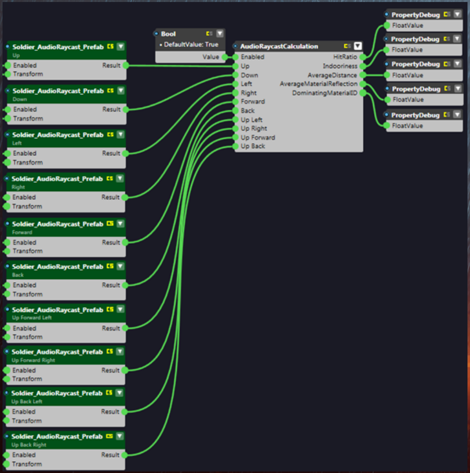
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ t \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix},$$

with the additional condition that,

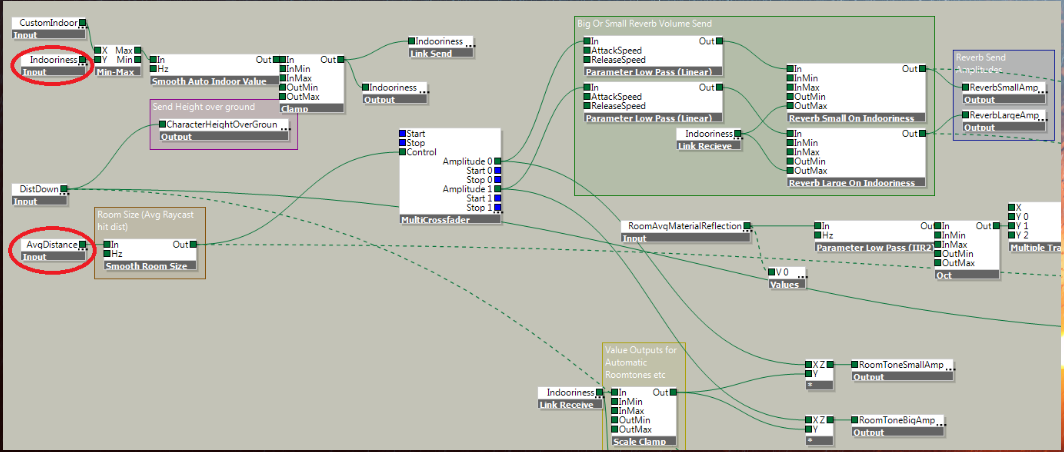
$$0 \leq t \leq 1, \quad 0 \leq u, \quad 0 \leq v, \quad u + v \leq 1,$$

must be satisfied for finite length lines ℓ and triangle surfaces Π.

Audio raycast schematics



Audio raycast sound patch



Physically based sky

Physically based sky

- *Physical based sky* informal term used by Gustav Bodare and Edvard Sandberg.
- Presented their results from their thesis *Efficient and Dynamic Atmospheric Scattering* at DICE's office.
- Objective: Render realistic atmosphere in real-time on GPU.
  - Simulation technique suitable for video games.
  - Implemented in Frostbite game engine.

Atmospheric scattering

Atmospheric scattering contributes to many everyday phenomena that we experience:

- Sky is blue at noon and red at sunset.
- Object far in the distance adapts the color of the sky.

Atmospheric scattering describes what happens with sunlight when it enters the atmosphere.

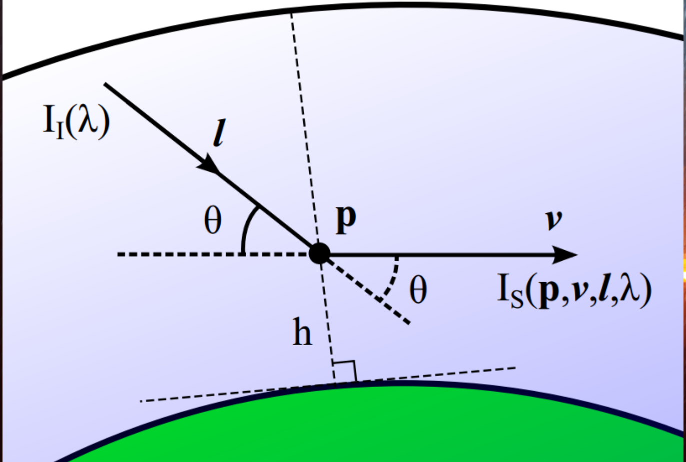
Scattering model

Incident sunlight I_I. Scattered sunlight I_S.

- Scatter divided into *Rayleigh* and *Mie* scattering.
- Rayleigh scattering (sub-wavelength particles)
- Mie scattering, larger objects (scattering and absorbing).

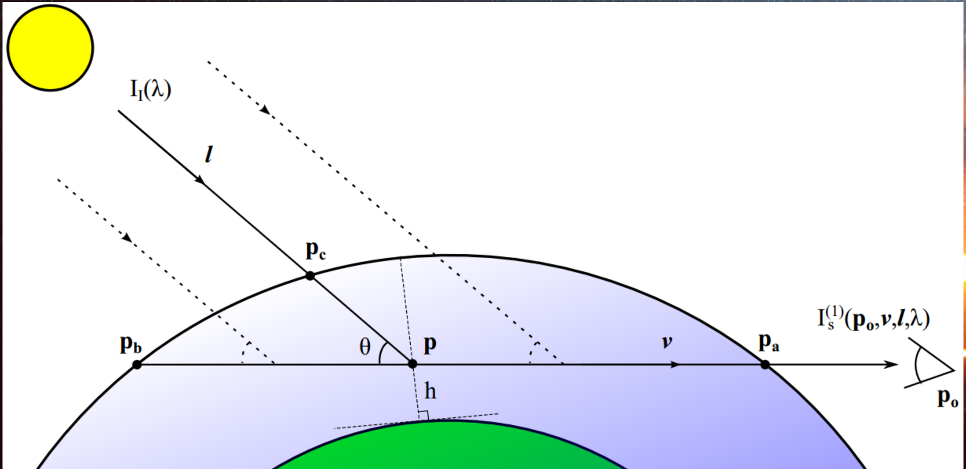
Scattering model (cont.)

$$I_S(\vec{p},\vec{v},\vec{l},\lambda) = I_I(\lambda)\left(\rho_R(h)F_R(\theta)\frac{\beta_R(\lambda)}{4\pi} + \rho_M(h)F_M(\theta)\frac{\beta_M(\lambda)}{4\pi}\right)$$



Scattering model (cont.)

$$I_s(\vec{p}_0, \vec{v}, \vec{l}, \lambda) = \int_\Gamma I_s(\vec{p}, \vec{v}, \vec{l}, \lambda) \cdot \mathrm{d}\vec{p}, \quad \Gamma(t) = (1-t)\vec{p}_b + t\vec{p}_a, \quad 0 \le t \le 1.$$



Precomputing scattering I_s

- One computation per color channel (R, G and B).
- 9 degrees of freedom per channel $\Rightarrow$ huge *lookup tables* (LUT).

Reduce the degrees of freedom down to:

1. Viewer height.
2. Viewer-zenith angle.
3. Sun-zenith angle.

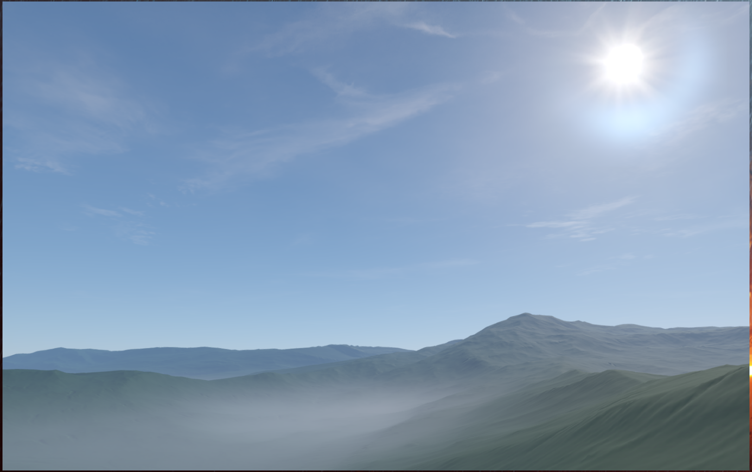
Assuming:

1. Planet is perfectly spherical.
2. Density of particles depends only on height.
3. Light reaching the atmosphere can be treated as parallel.
4. Removal of the sun-view azimuth angle (modelling assumption).

Implementation

- Multiple scattering computed recursively.
- I_s is discretized into three small 3-dimensional array.
  - Computed numerically using the trapezoid rule.
  - Stored on GPU as a 3D texture.
- Adaptive algorithms amortizing updates of LUT over multiple simulation ticks.
- Rendering sky $\Rightarrow$ sampling data from LUT; with special consideration for clouds.

Daytime



Sunset

Movie

`![PBS.mp4](video/PBS.mp4)`

Screen space reflections

Approximating reflections in real-time

- Real-time ray-tracing:
  - Accurate physical model for refraction, scattering, and dispersion.
  - Extremely costly to compute.
- Planar reflections:
  - Render scene excluding mirrored surface.
  - Applied recursively: Render and project on mirror surfaces.
- Environmental mapping:
  - Pre-computing technique, similar to above.
  - Render environment as texture on reflective object.
- ...

Screen space reflections

Screen space reflection (SSR) is a rendering buffer technique to render realistic reflections under certain constraints.

- Ray-tracing method, tracing in render buffers.
- Cost is independent of the complexity of the scene.
- Uses depth information (Z-buffer) and runs on GPU.
- Reflections limited to objects rendered on screen.
- Thesis by Mattias Johnsson:
  - Approximating ray traced reflections using screen space data

Algorithm (oversimplified)

1. Trace ray from source pixel in direction of its surface normal.
2. Ray is marched until it hits the end of the screen, or collides with an object.
3. A ray collides with a object if its depth value is behind the value of the object.
4. Mix colliding pixel color value with source pixel color based on the objects reflectivity.

Advanced AAA implementation

- Tech demo made by Yasin Uludag at EA DICE.
- *Hi-Z Screen-Space Cone-Traced Reflections* in *GPU Pro 5: Advanced Rendering Techniques* (CRC Press).

Results

```
![xbox_one_green.mp4](video/xbox_one_green.mp4)
```

Thank you for listening! / Questions?

- DICE has a close connection to academia.
  - Archive of technical reports: `http://dice.se/publications/`.
- Offers interesting thesis work (not only in programming):
  - Audio: Anders Clerwall: anders.clerwall@frostbite.com
  - Rendering: Yasin Uludag: yasin.uludag@dice.se.
- International career: `http://careers.ea.com/`