VISIBILITY AWARE STEERING WITH RASTERIZATION

JAMES ZHAO

Summary / Motivation

The purpose of this project is to create a natural looking and visibility aware 3D movement simulation, based on existing steering models. In steering simulations, an agent attempts to reach a target while avoiding a set of obstacles. Existing steering models aim to reproduce natural looking movement by iteratively modifying the agent's orientation. However, existing models treat obstacles as particles. They have no volume and thus are unable to occlude other obstacles. In this project, rasterization is used as a way to efficiently and accurately test occluder-agent visibility on a per-pixel basis.

Related Work

Exists steering models attempt to model human movement behavior by iteratively modifying the agent's heading.

Pre-existing steering models takes the agent position, agent orientation, target position, and obstacle positions as input (Warren, Fajen 2004). Their steering model consists of two terms: an obstacle repelling force and a target attraction force.

Target Attraction Term

$$- b\dot{\phi} - k_g (\phi - \psi_g)(e^{-c_1 d_g} + c_2)$$

This term attempts to modify the agent's orientation so that it is directly facing the target. The first subterm dulls any sudden orientation changes. The last subterm attenuates this term's power by the distance to the target.

Obstacle Avoidance Term

$$\sum_{i=1}^{\#\text{obstacles}} k_o (\phi - \psi_{o_i})\, e^{-c_3|\phi - \psi_{o_i}|} (e^{-c_4 d_{o_i}})$$

This term attempts to modify the agent's orientation so that it doesn't face the obstacle. The last two subterms attenuates this term's power by the current obstacle-agent distance and their orientation difference.

Simulation

Every simulation iteration, the terms in the above model are calculated and summed. Then, the orientation changes are applied and the simulation proceeds to the next iteration.

Visibility Testing

While the model above produces good looking results, it doesn't take into obstacle visibility. It treats every obstacle as a particle with no volume. I attempted to use rasterization to compute visibility.

Approach

Rasterizing the scene from the agent's point of view provided information on what obstacles are visible.

Textures

Depth Texture (R32F) - Screen sized

2x Solution / Intermediate Render Targets (R16G16B16A16F) - Screen sized

Cpu Readable Render Target 1x1

Procedure

Clear Depth Texture to -1

First I cleared the depth texture to -1. This default value denotes that a particular pixel is empty. Note: This depth texture is a render target and is not the depth target used for depth testing.

Render Linear Depth

I rendered depth using a pass-through vertex/pixel shader. Depth is scaled to the (0, 1) range for easy visualization (z/zFar).

Compute the obstacle term in a per-pixel fashion using the rendered depth

For every pixel,

1. If the pixel is empty (z = -1), ignore it
2. Determine the heading difference (How far off-center is it?)
3. Plug it into the obstacle term and write the result to the Solution Render Target

Downsample the Solution Render Target to 1x1

I used two intermediary textures and ping-pong downsampled (until 1x1 size) between them. Empty areas (where the obstacle term = 0) are ignored in the averaging process)

Read the 1x1 value back to the CPU

The readback using the DirectX API required a special 1x1 staging texture to be created.

Calculate the Target Attraction term on the CPU

Sum the Obstacle Avoidance Term and the Target Attraction Term, and apply it to the agent

Challenges

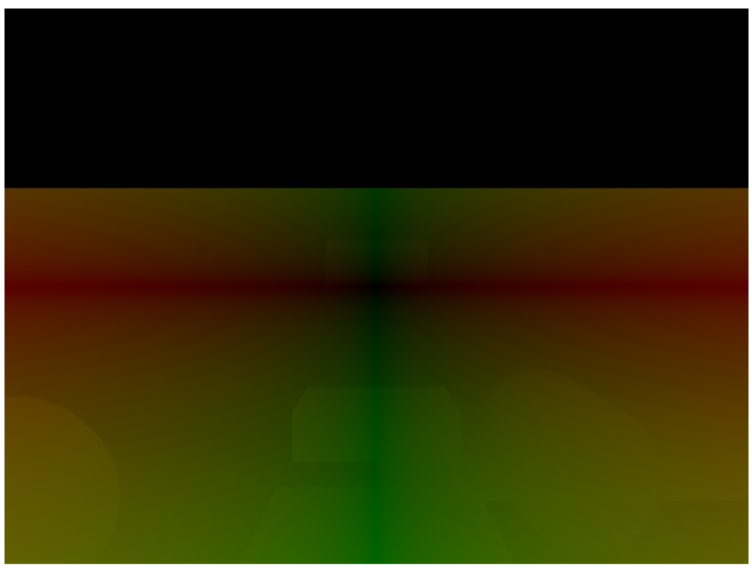
Reading back 16 bit floats

It took roughly 45 minutes to figure out how to decode GPU-style half floats into full 32 bit floats. GPU-style floats, converted to binary, didn't seem to follow the traditional sign-exponent-mantissa pattern (most likely it was in some weird order). In the end, I found out the graphics API provided a function for the conversion.

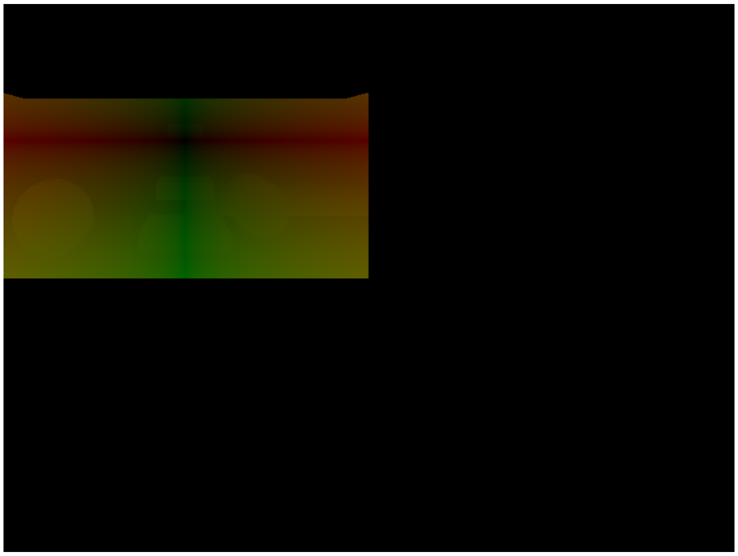
Equilibrium Cases

When obstacles appear roughly equally on all sides of the view direction, the summed obstacle avoidance term approaches 0. Thus, the agent appears to steer straight into the obstacle.

Results / Debug Visualizations



Obstacle Avoidance Term Visualization



Downsampling - 1st Iteration

Linear Depth Visualization

Conclusion

Traditional steering models efficiently simulates natural-looking movement using infinitely small obstacles and targets. Using particle style markers introduce issues as there's no occlusions of any sort. Using rasterization, we can efficiently test visibility of obstacles from the agent's point of view while using the traditional steering model.

Bibliography

Fajen, Warren. 2004. From Optic Flow to Laws of Control

Fajen, Warren. 2003. Behavioral Dynamics of Steering, Obstacle Avoidance, and Route Selection

Fajen, Warren. 2007. Behavioral dynamics of intercepting a moving target