Dealing with performance issues

Normally when a map is well designed with the game's limitations in mind, the game does a good job of optimizing the environment through the use of occlusion culling. However, there may be times where it appears to take into consideration objects that are not visible to the player. Of course, if the level designer (and scripter) does not fully consider what can decrease performance, then there isn't much that the game can do to optimize. This article will touch on mostly (flawed) map and scripting design, as there is nothing us users can do about the game itself.

Keep track of your objects

The main cause for performance issues that i have found deals with having too many objects with high poly counts. There is an apparent limit as to how many objects with a high poly count that the user can have near him (though this may be more dependent on the hardware). At a far enough distance they tend to be okay, but if the user gets at a certain distance from the objects, the game may decide to take them into consideration even if they are not visible to the user.

Physics

If there is an extreme performance drop at the start of a map, it is most likely caused by entities falling. For low-poly, non-organic types, this is usually not a concern, but things like corpses the collision and gravity will drain performance.

Make use of corridors

Making use of corridors (i.e. mostly L-shaped) prevents the user from looking too far. This gives a chance for occlusion culling to work its magic. Even if you have doors blocking the way between each room, you should still consider corridors, as not all combinations of doors and door frames properly block out the user's view (though the player may not be able to physically see behind it).

Make use of global fog

You can activate the global fog in the level settings (Edit > Level settings > Fog) in the Level Editor. The global fog can be used to prevent the game from rendering objects beyond the ending point of the fog if culling is enabled for the global fog. For indoors it should be safe to allow for a longer end point, but use what you find best for your story.

The fog distance can be set so that the fog is never seen by the player, but it still culls a lot of the map. For example, if the longest corridor of the map is 12 meters long, setting the fog distance to 14 should work. Just in case the player sees it, however, set the fog's colour to black; this way, it will look like darkness.

Reduce particle effects

Having too many particle effects can reduce performance. This includes lamp entities, be they torches, candles or otherwise. If you require more light, consider simply adding dimmed box lights or point lights with a large radius or replacing the wall with a window, or have a sunroof, with a spot light shining through.

Use fading on particle effects

Having a big number of particle effects is doable in HPL2, but they need to fade at distance (note: if you are already using fog culling, this probably won't help much). When placing a particle in the map, set it's fade distance so that the player can't see it when they are too far. Depending on your map, you might want to fade in a bigger particle in the place of few small ones - this way something will be visible from distance, but still will increase performance.

The more particles there are in the map, the more this method will be effective. From the author's experience, this allows the map to have a couple hundred particles in the map without any performance drop.

Example: Outside map with rain. There are two particles needed to be placed every couple meters: splashes on the ground and the actual rain. These particles need to cover the whole area, so there will be a lot of them and they'll decrease the performance significantly. However, the player will only see only the nearest 1 to 4 of these particles at once; therefore the rest of particles can fade at e.g. 10 meters from the player.

Avoid fast-looping timers

Having a function loop too fast by use of timers can cause performance issues. Anything below 10ms is pushing it. Ideally, you'd want at least 20ms. That should be more than enough for any event.

IMPORTANT: Timers in Amnesia (along with all forms of callbacks) are indexed in a 16-bit integer value for every session, meaning the maximum that can be triggered is about 65,535. If you use a lot of fast-looping timers, avoid exceeding this number of total calls. The fastest a timer can loop seems to be the tickrate of the game, which is 60/s (16ms), so by making a timer that loops this frequently, the index would run out of bounds in approximately 15 minutes. Once this limit has been reached, the game will crash with a BlackBox error upon the next loading screen. By shutting down and restarting the game, this index is refreshed, so it does not apply to save loads, only the current session. This is however not a solution to the issue and not a desirable effect to place upon the player.

Tips

Spotting "leaks" in walls

Though the player may not notice holes or cracks in between each wall, the game most likely will.

One method you could use to find these cracks is to have all the lights turned off and a white background behind the walls (e.g. a completely white skybox), or have the room in question completely dark and the surrounding halls and rooms full of light. Then, as you navigate through your map, you'd be able to easily catch any leaks that need filling.

Notes

This article is a work in progress. More will be added as they are discovered.

For players that aren't part of the story creation process, turning off World Reflection can help increase performance.

The game uses OpenGL, which allows for a cross-platform experience. However, the drivers for your graphics card may not be well optimized for OpenGL (since DirectX is more popular).