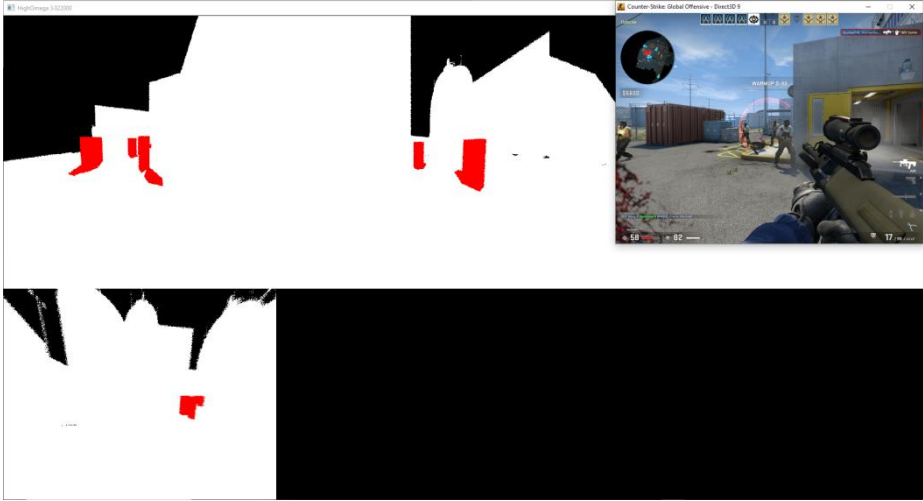


SauRay™: A Decisive Blow to Wallhacks!



What is SauRay™?

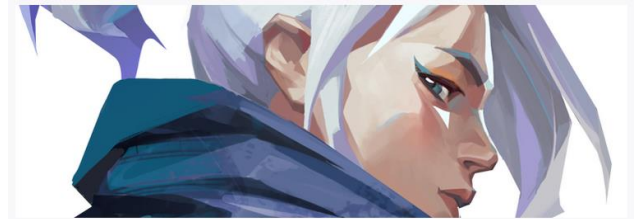
Server-side hardware-raytracing
antiwallhack solution:

- Middleware that runs on a game server that has a raytracing GPU
 - Renders a visibility focused version of each client frustum at low resolution in a smart manner:
 - Accounts for client-side rendering pipeline effects like shadows, global reflections or global AO.
 - Creates a visibility matrix that outlines who can see who.
- 
- Provides this matrix to the game server loop to filter outgoing updates!
 - If player A can't see player B, player B's updates are not transmitted to player A.



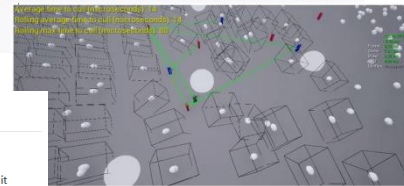
Prior work

- Valorant's Fog of War:
 - Too many false negatives
 - Abandoned in favour of floodfill PVS
- CornerCulling:
 - Community effort by Andrew Huang + Robert @ CC:
 - Anticheat @ Ubi on R6:Siege
 - Largely unmaintained
- SourceMod Anticheat (SMAC):
 - Source games only
 - Unmaintained
 - Same issues as above
- Some paywalled stuff from Russia?
 - Source games only and...
 - ...Not much better

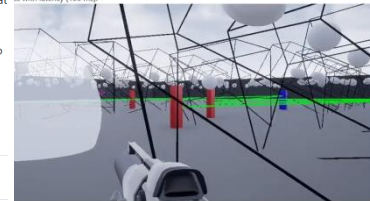


APR 14, 2020

DEMOLISHING WALLHACKS WITH VALORANT'S FOG OF WAR



is with latency (100 ms)



SMAC

SourceMod Anti-Cheat

For a lot of people, SMAC has been one of those more elusive plugins due to some of the issues surrounding it involving copyrights and headers. In 0.8.6.0, the original authors added the headers to it and left it at that. During that time, I was working on my own fork of it specifically for ZPS which evolved from there into the current fork that is seen today. So not only does the code have all the appropriate headers, but included is a license with those headers as well, so it should be okay to distribute, post, branch, and fork once again as needed provided everyone adheres to the license.

For information about the plugin and its modules, please use the wiki here: <https://github.com/Silencio/SMAC/wiki>

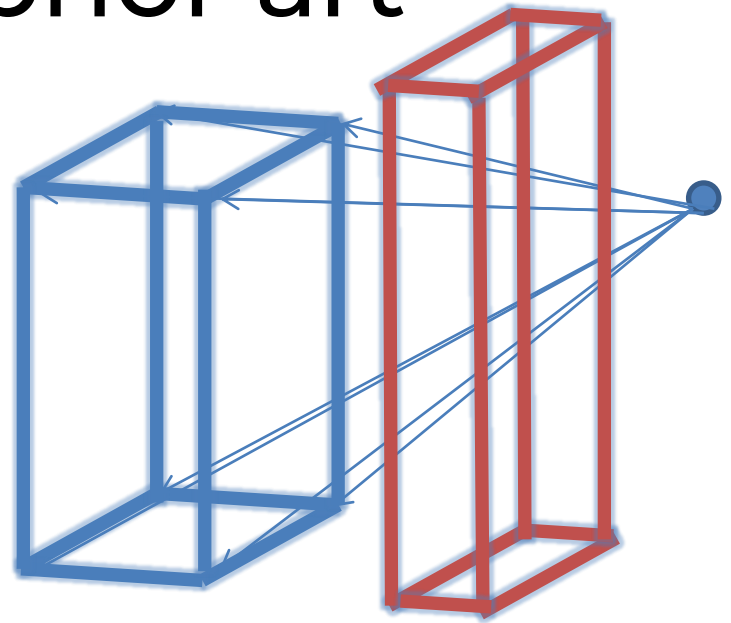
Changelog

0.8.7.3 Update (12-20-2019)



Issues with prior art

- Extremely tight ray budget
 - All CPU-side
- In some cases crude/conservative representation of scenery
 - Tons of false negatives/popping
- At most checked only 8 corners on any given player
 - Nowhere near enough
- Could not account for shadows, reflections, global AO in case they're there
 - Also could not account for further complicated scenarios like a shadow in a mirror through a tiny hole in the wall



- **Verdict?**

Not enough horsepower to solve the problem



Our solution

Just give the server what it needs!



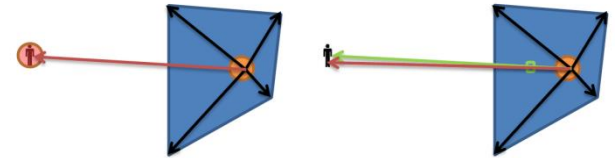
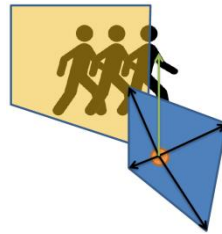
Problem solved right?



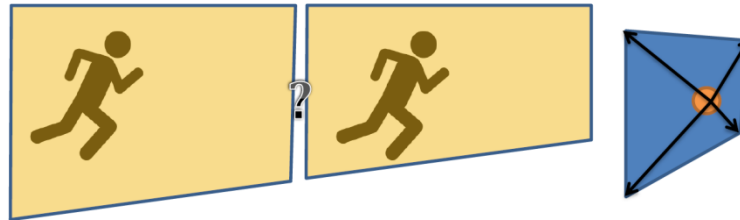
Not quite...

- Ray budgets while higher, still not infinite
- Need to use graphics techniques like:
 - Selective super-sampling

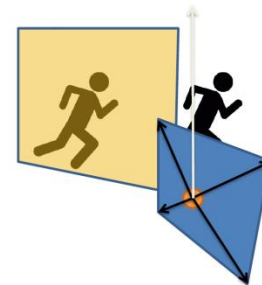
- Forward projection of player geometry:



- People might be running fast
 - and/or low tickrate!!



- SauRay™ may be behind!
 - Parallel execution to game loop for performance
 - Low tickrate further necessitates!!

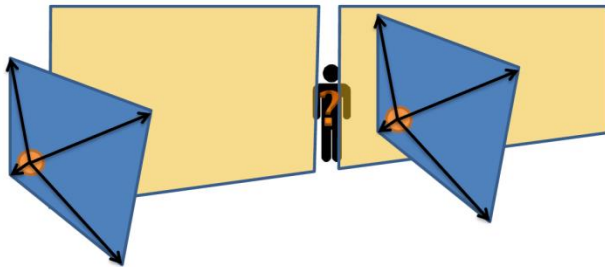


(Projection) doubles for viewers too!

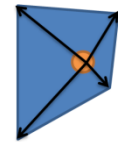
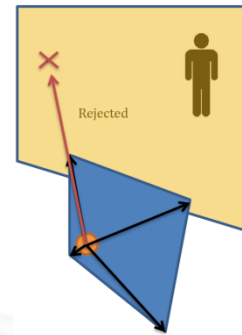
- Viewers also need to peek into the future for all the same reasons



- Fast movement and/or low tickrate



- SauRay™ may be behind
- Low tickrate further necessitates!



Full details are in our tech brief!

Visit <http://sauray.tech>

- Read our F.A.Q
- And our technical brief!
Only 35 pages ;)
- We mainly target low-latency/competitive scenarios
- Though our brief also discusses our strategies for very high latencies as well (+100ms RTT)



[HOME](#) [DEMONSTRATION](#) [APPEARANCES](#) [PORTFOLIO](#) [PARTNERS](#) [ABOUT](#) [F.A.Q](#) [CONTACT](#)

F.A.Q

HOW DOES IT WORK?

Our solution works by ray-tracing the frusta of all players simultaneously on the server with every tick. SauRay's specially crafted algorithm makes careful use of a limited ray budget to solve an extremely challenging dynamic visibility problem for all virtual frusta at low resolution. It requires hardware ray-tracing capable GPUs on the game server and can be extended to do software ray-tracing if necessary (i.e. if the workload is not triangle and/or BVH based).

Our [patent application](#) can be found here.

We have made our [technical brief](#) publicly available as well.



Fun fact #1: traffic goes down!

- So we did some traffic analysis:
 - Client traffic on competitive-style matches: 2 runs, 2 sessions per run (T&CT)
 - CS:GO, de_nuke, 5v5
- Adjusting for total session length differences (~3 minutes), **we save bandwidth** in absolute terms!

Packets A → B	Bytes A → B	Packets A → B	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
86,494	<u>36.856 MiB</u>	85,180	<u>15.778 MiB</u>	0.000000	<u>1356.4368</u>	222.585 KiB	95.289 KiB
1	67 bytes	1	60 bytes	533.402120	0.0165	31.734 KiB	28.419 KiB
2	138 bytes	2	257 bytes	533.407438	0.0276	39.129 KiB	72.870 KiB

Packets A → B	Bytes A → B	Packets A → B	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
99,153	<u>54.038 MiB</u>	98,282	<u>18.148 MiB</u>	0.000000	<u>1545.7026</u>	286.392 KiB	96.183 KiB

$$\left(1.0 - \frac{36.856}{\frac{1356.4368}{1545.7026} \times 54.038} \right) \times 100.0 =$$

22.279%

- Turns out player updates are chunky:
 - Contain animation data
 - Quake/Euler angles – legacy of idTech2
 - Player loadout etc.
- This is while having to add (randomized) origins for transmitted gun/walk sounds as player entity locations maybe withheld



Fun fact #2: we found a bug in CS:GO!

- Turns out CS:GO re-uses visibility leaf filtering for client-side occlusion culling too!
 - It only accounts for the player AABB and not the gun
- In rare configurations the gun can go missing
- Just relying on SauRay™ serverside filtering would prevent this! ;)



Cost: (perf. and \$\$\$)

Device	CSGO Avg ms.	TF2 Avg ms.	Server Type	# of 64-tick servers	NRE (approx. CAD)	Running cost (Annual)	Running cost (Annual 1-yr ctrct.)	Running cost (Annual 3-yr ctrct.)
RTX 2080Ti	4.95	6.17	Commodity	2	\$1,000	471.08	N/A	N/A
T4 Turing (Close to RTX 2070 Super)	6.17	7.39	Cloud - AWS G4dn.xlarge	2	N/A	6,306.87	3,788.92	2,517.95
RTX 2060 OC	7.96	9.65	Commodity	1	\$500	308.62	N/A	N/A
RTX 3070Ti (Laptop!)	7.94	7.94	Commodity (Laptop!)	1	\$1,500	376.22	N/A	N/A



Netcode architecture: do's and don'ts!

Do:

- Implement thin-client/server-authoritative architecture
 - We can deal with its absence... kinda. Better to have.
- Scrutinize user-supplied data (y-FOV, aspect ratio etc.)
- Use non-conservative geometry simplification
 - Get rid of small occluders...
- Be mindful of side-channels:
 - Obfuscate/randomize origins, stream audio etc.
 - Trace projectiles, gunshot point light spheres etc.

Don't:

- Assume player data to be always available
 - Pack origin when necessary
- Send more data than necessary
 - Radar pins don't need 3D locations



We have stuff to show!

We have Paris servers running until Monday! (AWS eu-west-3)
(Afterwards) you can visit our US West Coast servers! (AWS us-west-2)

Running until further notice...

Details are on Twitter: [@antiwallhack](#) and our website.

NOTE: **We don't advocate breaking T.O.S**

However, *if* you know what you're doing, come and test our schtuff! ;)



Thank you!



Time for
demo and Q&A!

