GDC

The Elusive Frame Timing

A Case for Smoothness Over Speed

Alen Ladavac
CTO
Croteam

<taxonomy>

“perfect”

[Video] Perfect

“slow”

[Video] Slow

“spikes”

[Video] Spikes

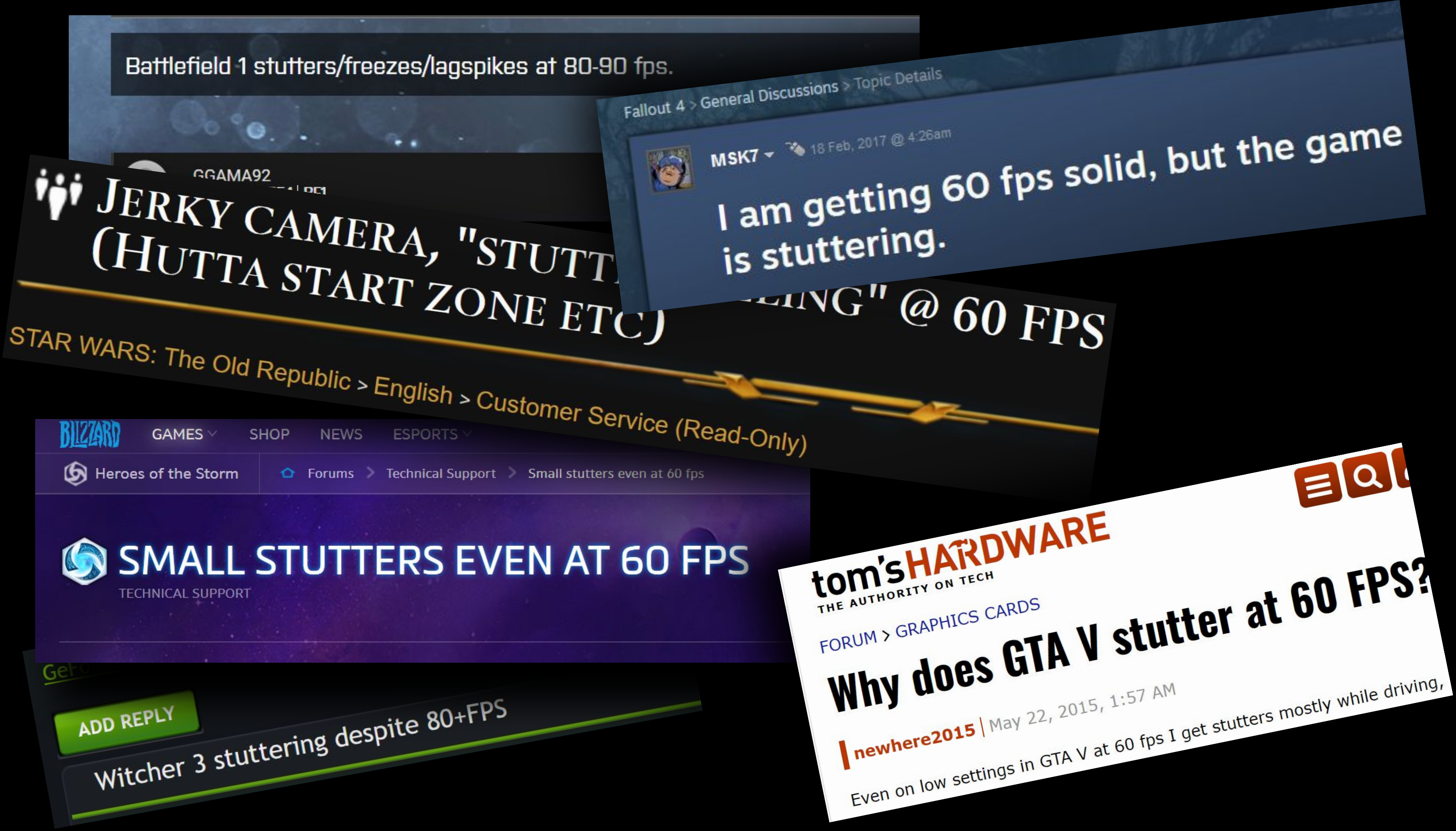
“jagged”

[Video] Jagged

“heartbeat”

[Video] Heartbeat

just a bug?



Battlefield 1 stutters/freezes/lagspikes at 80-90 fps.

GGAMA92

Fallout 4 > General Discussions > Topic Details

MSK7 18 Feb, 2017 @ 4:26am

I am getting 60 fps solid, but the game is stuttering.

JERKY CAMERA, "STUTT … ING" @ 60 FPS (HUTTA START ZONE ETC)

STAR WARS: The Old Republic > English > Customer Service (Read-Only)

BLIZZARD GAMES SHOP NEWS ESPORTS

Heroes of the Storm Forums > Technical Support > Small stutters even at 60 fps

SMALL STUTTERS EVEN AT 60 FPS

TECHNICAL SUPPORT

ADD REPLY

Witcher 3 stuttering despite 80+FPS

tom's HARDWARE THE AUTHORITY ON TECH

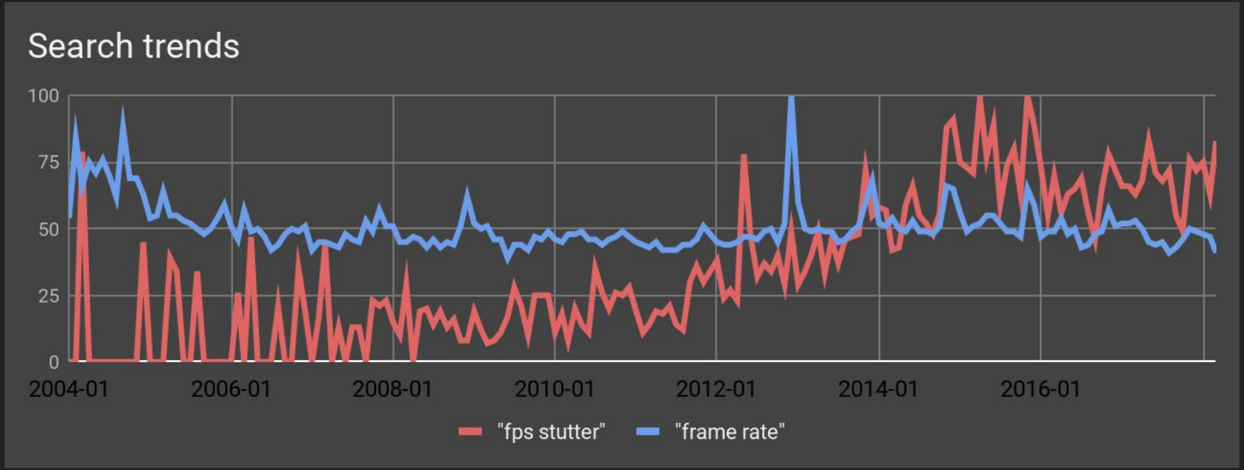
FORUM > GRAPHICS CARDS

Why does GTA V stutter at 60 FPS?

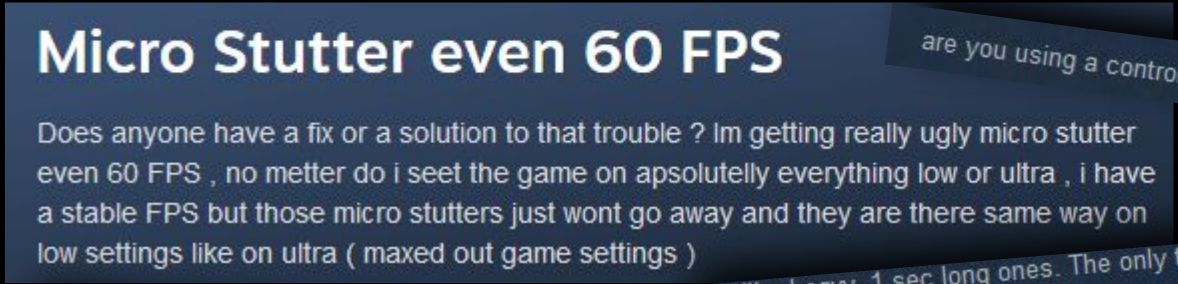
newhere2015 | May 22, 2015, 1:57 AM

Even on low settings in GTA V at 60 fps I get stutters mostly while driving,

Smoothness vs Speed

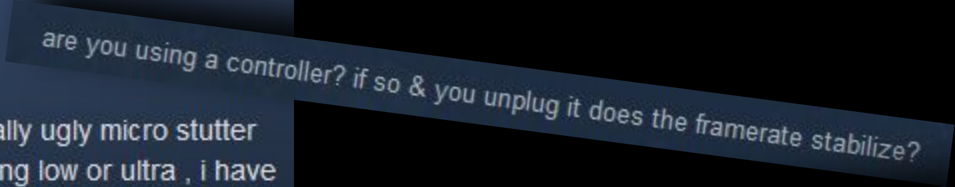


In the past 5 years, stutter is a bigger problem than performance!

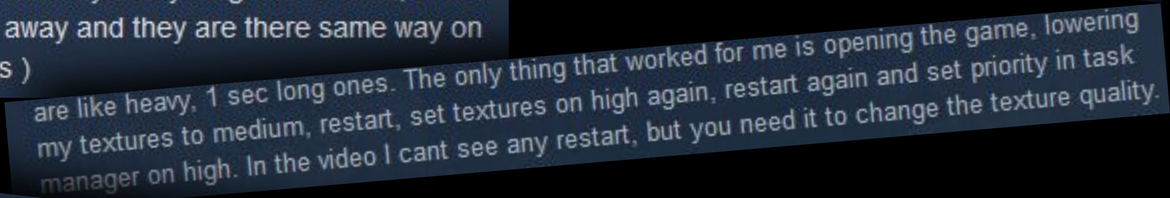


Micro Stutter even 60 FPS

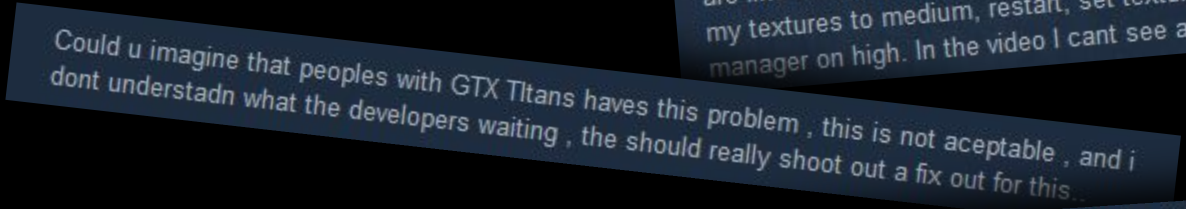
Does anyone have a fix or a solution to that trouble ? Im getting really ugly micro stutter even 60 FPS , no metter do i seet the game on apsolutelly everything low or ultra , i have a stable FPS but those micro stutters just wont go away and they are there same way on low settings like on ultra (maxed out game settings)



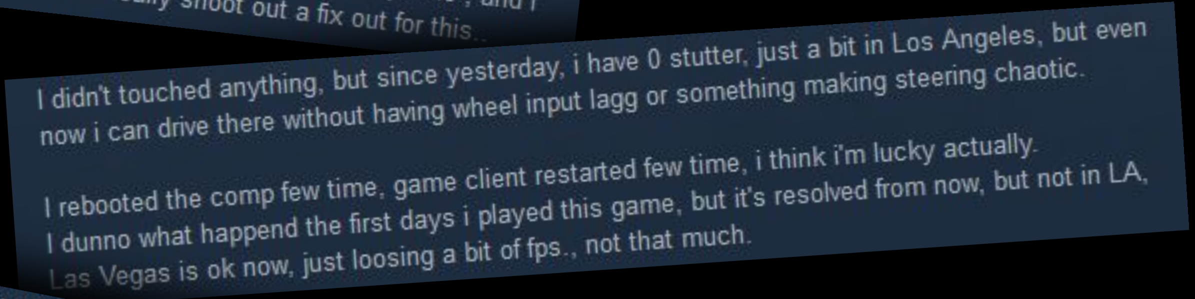
are you using a controller? if so & you unplug it does the framerate stabilize?



are like heavy, 1 sec long ones. The only thing that worked for me is opening the game, lowering my textures to medium, restart, set textures on high again, restart again and set priority in task manager on high. In the video I cant see any restart, but you need it to change the texture quality.

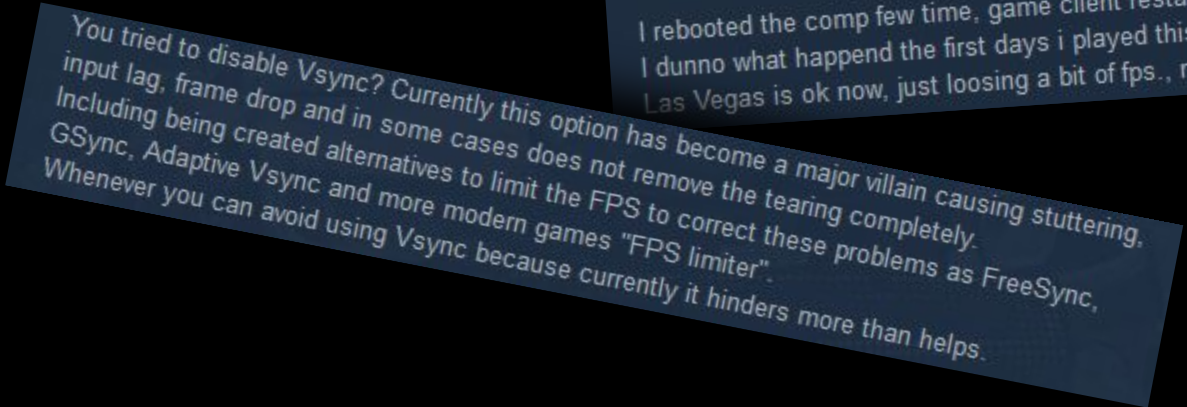


Could u imagine that peoples with GTX TItans haves this problem , this is not aceptable , and i dont understadn what the developers waiting , the should really shoot out a fix out for this..

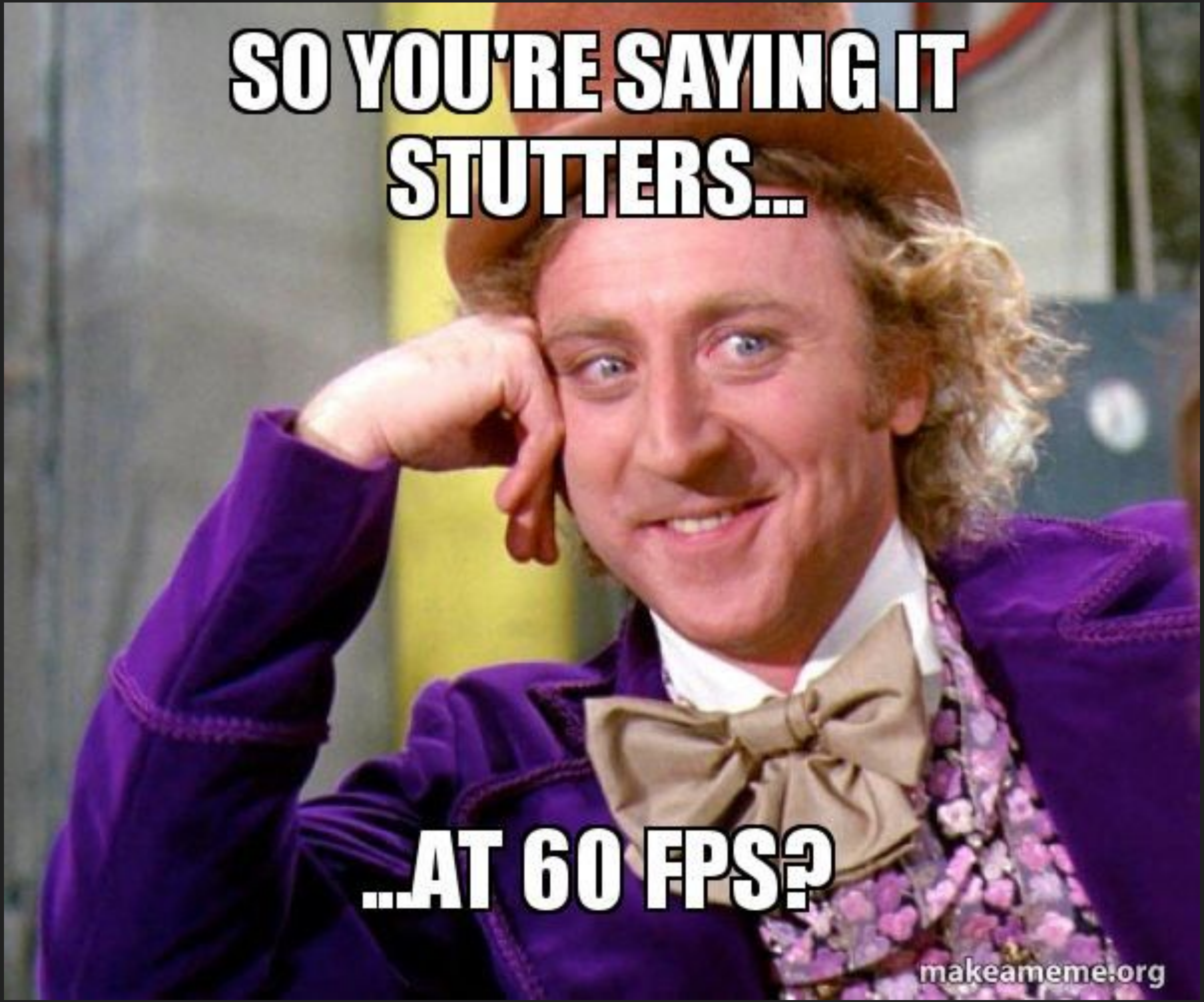


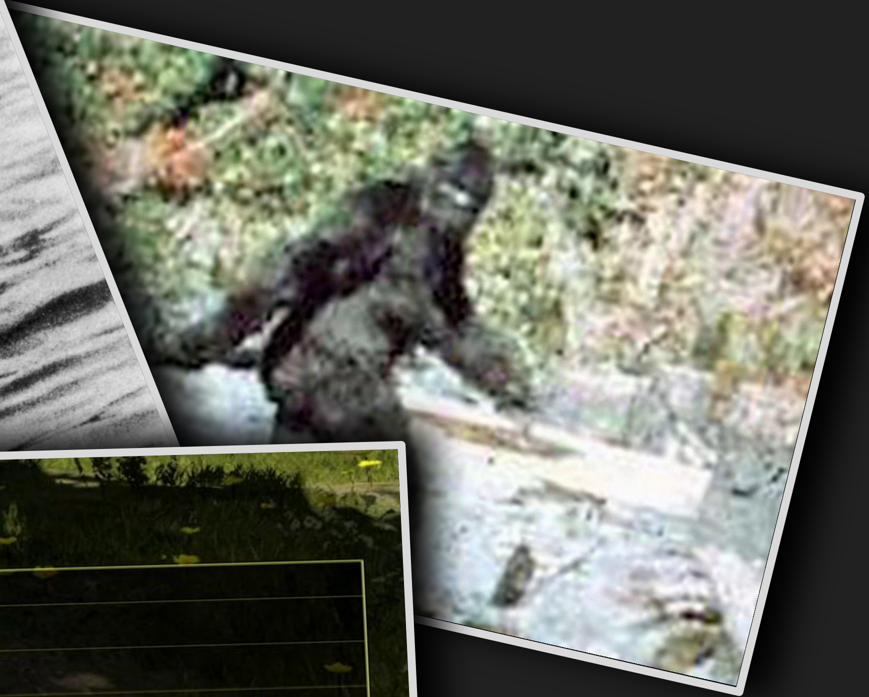
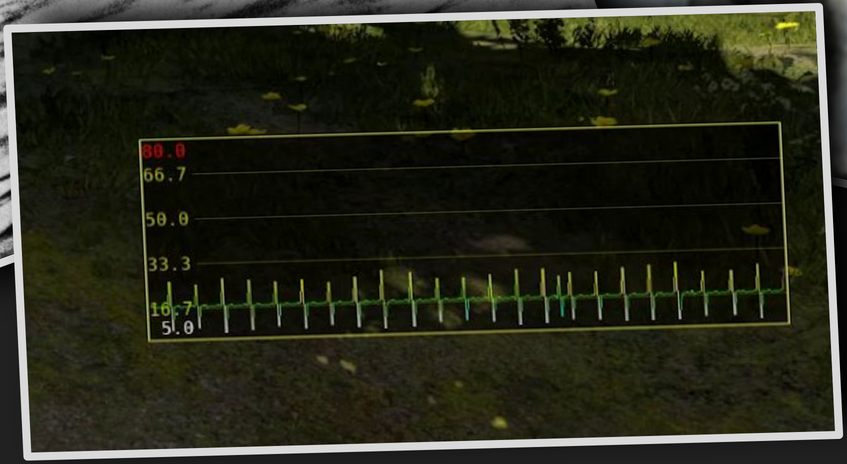
I didn't touched anything, but since yesterday, i have 0 stutter, just a bit in Los Angeles, but even now i can drive there without having wheel input lagg or something making steering chaotic.

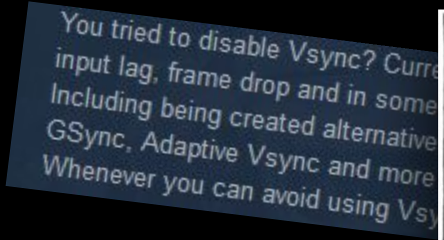
I rebooted the comp few time, game client restarted few time, i think i'm lucky actually.
I dunno what happend the first days i played this game, but it's resolved from now, but not in LA,
Las Vegas is ok now, just loosing a bit of fps., not that much.



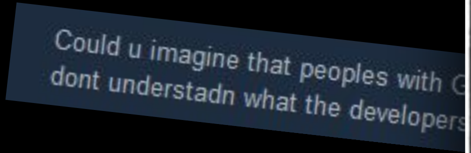
You tried to disable Vsync? Currently this option has become a major villain causing stuttering, input lag, frame drop and in some cases does not remove the tearing completely.
Including being created alternatives to limit the FPS to correct these problems as FreeSync, GSync, Adaptive Vsync and more modern games "FPS limiter".
Whenever you can avoid using Vsync because currently it hinders more than helps.


SO YOU'RE SAYING IT STUTTERS...
...AT 60 FPS?
makeameme.org



80.0
66.7
50.0
33.3
16.7
5.0



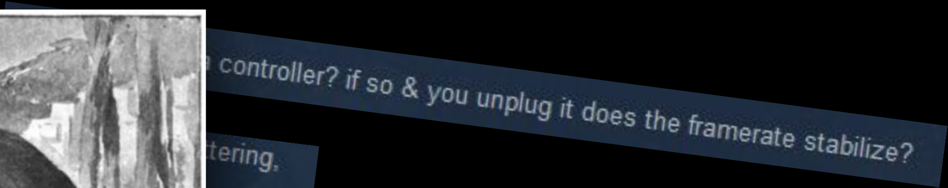
You tried to disable Vsync? Curr
input lag, frame drop and in some
Including being created alternative
GSync, Adaptive Vsync and more
Whenever you can avoid using Vsy



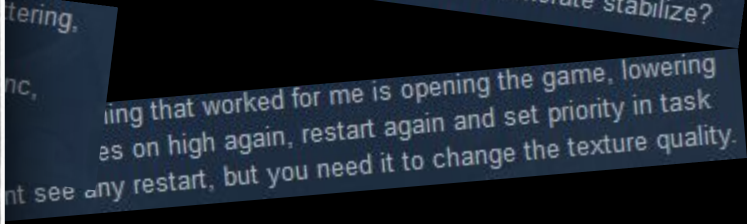
Could u imagine that peoples with G
dont understadn what the developers

Micro Stutter ev

Does anyone have a fix or a solutio
even 60 FPS , no metter do i seet
a stable FPS but those micro stut
low settings like on ultra (maxed



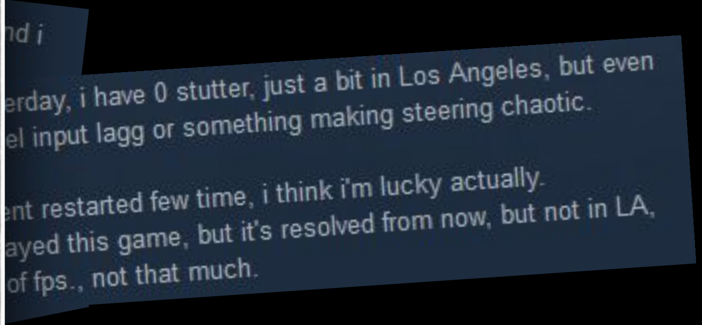
controller? if so & you unplug it does the framerate stabilize?



tering,

nc,

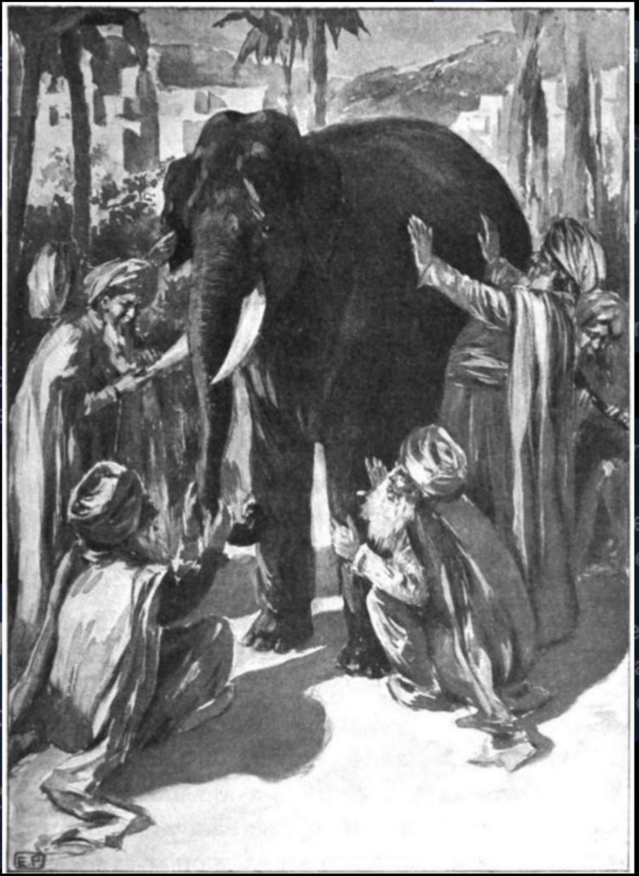
ing that worked for me is opening the game, lowering
es on high again, restart again and set priority in task
nt see any restart, but you need it to change the texture quality.



nd i

erday, i have 0 stutter, just a bit in Los Angeles, but even
el input lagg or something making steering chaotic.

ent restarted few time, i think i'm lucky actually.
ayed this game, but it's resolved from now, but not in LA,
of fps., not that much.



The Parable of Blind Men and an Elephant

frameology

(n.) The scientific study of the behaviour, structure, physiology, classification, and distribution of frame stuttering.

[Video] Heartbeat to Perfect

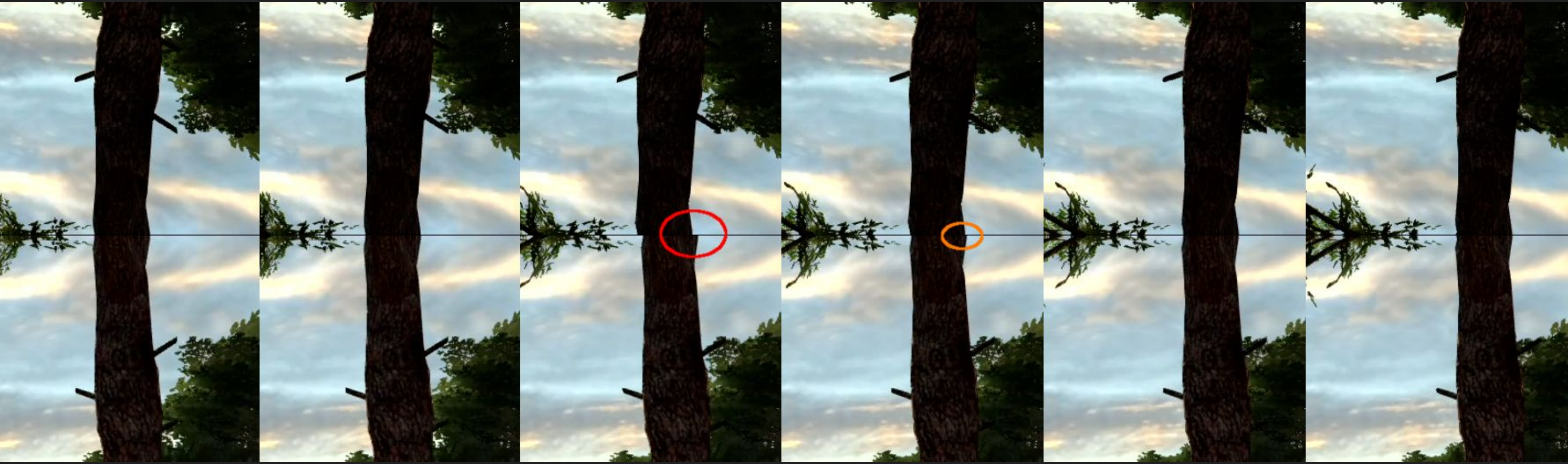
what’s the secret?

[Video] Heartbeat vs Perfect - split

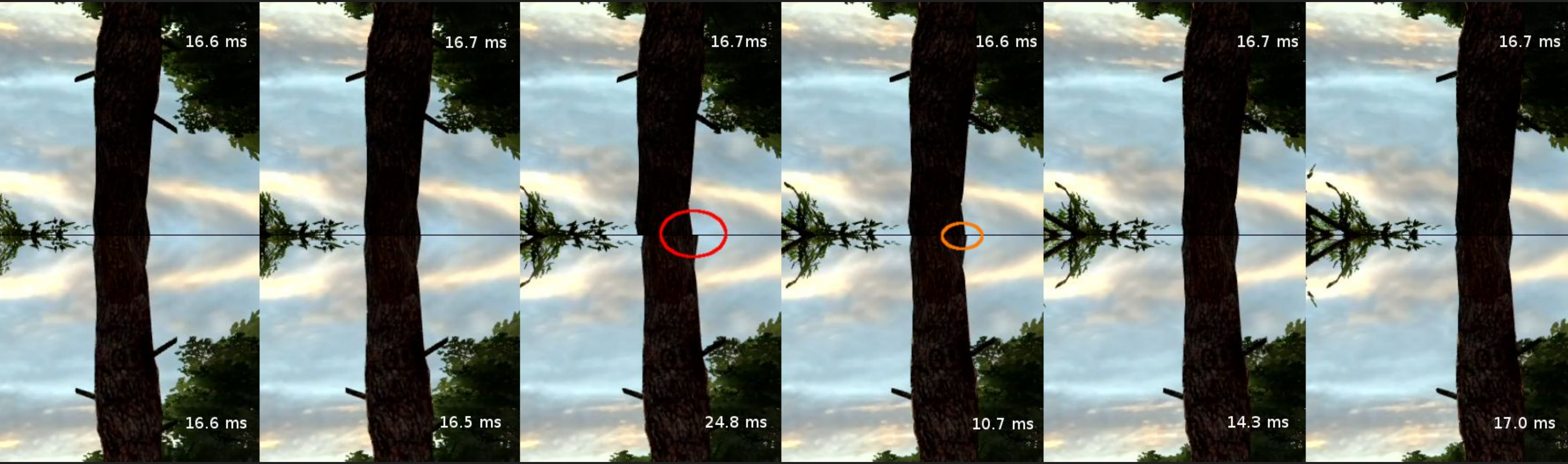
are you sure it’s
not skipping frames?

[Video] Heartbeat vs Perfect - split slow what was that?

The stuttering case is “faster” than the perfect case?!?!



Actually, it’s the opposite!



Stutter happens because the game <u>doesn’t know</u> how fast it is displaying!

The Secret

- “Dear game, pretend that this is 60 FPS.”
- Magic!
- Because it always *was* running perfectly anyway!

[Video] Heartbeat to Perfect (again)

Why does the game “think” it is running slower?

- 80’s/90’s - 8-bit/16-bit era
  - Fixed hardware, always same timing - no problems.
  - Remember the different NTSC/PAL versions?
- ‘90’s/00’s - software rendering/”graphics accelerators”
  - Started doing timing and interpolation
  - But no pipelining - no problem.
- What’s going on today?
  - Don’t know exactly but...

Theory: “API’s/Driver’s fault”*

- the real GPU load is “hidden” from the game
- The “feature” *possibly* introduced by “driver benchmarking wars” in early 2000s
  - Indicative by Flush() and Finish() behavior changing at that time
- Compositors are not helping either
- Internal mechanisms that are trying to compensate for this?
  - **That’s why this was so hard to find!**
- Inevitable anyway - to use pipelined hardware to its potential
- It’s OK to “buffer the slack time” - but <u>we need to know!</u>

* Largely speculation. :)

The two faces of wrong timing:

#1 - Wrong timing feedback

- Major cause of “heartbeat” stutter (but also sometimes others).

#2 - Wrong frame scheduling

- Major cause of stutter when recovering from “slow” to “perfect”

Proposal

- Must know how long a past frame lasted.
  - Asynchronous.
  - Will need to use heuristics
  - Must accept that it is not perfect.
    - Ideally, know how long the **next** frame will last. But that's not possible.
      - Or is it???
- Must be able to schedule when the next frame is shown.
  - *Faster is not always better!*
- Must know how much leeway we have left.
  - This is actually the most problematic part.

The old algorithm

```
frame_step = 16.67 ms // (assuming 60fps as initial baseline)
current_time = 0
while(running)
   Simulate(frame_step) // calculate inputs/physics/animation... using this delta
   RenderFrame()
   current_time += frame_step
   PresentFrame()// scheduled by the driver/OS[1]
   frame_step = LengthOfThisFrame() // calculated by the game[2]
```

[1]Who basically doesn’t have a clue.

[2]Who basically doesn’t have a clue.

The new algorithm

```
frame_step = 16.67 ms // (assuming 60fps as initial baseline)
current_time = 0
pending_frames_queue = {} // (empty)
frame_timing_history = {}
while(running)
   Simulate(frame_step) // calculate inputs/physics/animation... using this delta
   RenderFrame()
   current_time += frame_step
   current_frame_id = PresentFrame(current_time)
   AddToList(pending_frames_queue,  current_frame_id)
   QueryFrameInfos(pending_frames_queue,frame_timing_history)
   frame_step = FrameTimingHeuristics(pending_frames_queue, frame_timing_history)
```

```
legend:
 New APIs
 New App Algorithm
```

Internals of `FrameTimingHeuristics()`

- Poll all in `pending_frames_queue` - for those that are already available
  - record their respective timings into the `frame_timing_history`.
- If you see any single frame that missed its schedule,
  - Return its length to be used for `frame_step`
    - (this is how we drop into lower framerate!)
- If you see `recovery_count_threshold`* successive frames that are both
    - Early *and*
    - Their `margin < recovery_margin_threshold`*
  - Return their length to be used for `frame_step`
    - (this is how we bump into higher framerate

***`recovery_count_threshold` *and* `recovery_margin_threshold` *assure we don't start oscillating up/down*

OpenGL + VDPAU prototype

- Implemented in The Talos Principle as proof of concept in Aug 2015
- Uses NV_present_video OpenGL extension
  - Originally intended for video playback - thus has timing features
- Almost there:
  - Properly schedule future frames
  - Get timing info for past frames
  - But no margin info
    - Makes it very hard to recover
- Only works on some NVIDIA boards, on Linux, under OpenGL
  - Not very wide coverage, but proved the point

Vulkan + VK_GOOGLE_display_timing

- Implemented in The Talos Principle and Serious Sam Fusion...
  - .... just in time for this talk
- Has everything:
  - schedule future frames
  - timing info for past frames
  - has margin info
    - ambiguity?

the results

remember the “spikes”?

[Video] Spikes w/GDT

“jagged”?

[Video] Jagged w/GDT

“heartbeat”?

[Video] Heartbeat w/GDT

“heartbeat”

has turned into

“perfect”!

20 FPS???

- `”Acceptable frame-rates in GLQuake begin at 20 FPS, and 25 FPS for GLQuakeWorld”`

  (from Comparison of Frame-rates in GLQuake Using Voodoo & Voodoo 2 3D Cards, by "Flying Penguin (Mercenary)" cca year 1999.)

  - In those days spirits were brave, the stakes were high, ... and framerates sucked???
  - Was the tolerance really that low?
  - Probably, but also - ***those* 20 FPS were certainly smoother than “*today’s*” 20 FPS!**

check this out...

[Video] 20 FPS w/GDT

Internals of `FrameTimingHeuristics()`

- Poll all in `pending_frames_queue` - for those that are
  - record their respective timings into the `frame`
- If you see any single frame that
  - Return its length to be u
    - (this is h)
- If you se successive frames that are both
  - `< recovery_margin_threshold*`
  - their length to be used for `frame_step`
    - (this is how we bump into higher framerate

But don’t stick to this!

*`recovery_count_threshold` *and* `recovery_margin_threshold` *assure we don't start oscillating up/down*

Considerations

- When to decide to recover from slow back to perfect?
- How (and whether?) to correct for timing when dropping to slow?
- Can we predict slow and make a perfect drop?
  - This would be the “Holy Grail” of smoothness - when possible.
- Can probably do even better than this!
- What about VRR displays? What if Vsync is off?
  - That’s both doable - but still no API for it yet!

Subjectivity

- It *is* a matter of perception in the end
- Perhaps some people see it differently
- Different developers will have different approaches
- Expose different options to users?

Platform support...

- VK_GOOGLE_display_timing was defined Mar 2017 , but...
  - Only Android - only Shield TV and a handful others
  - As of last month, available on Linux in RADV driver as part of Mesa!
  - Everyone else - Please implement ASAP! 😊
- DirectX 12?
- Metal?

Does this always apply?

- *Only* if FPS can fall below refresh rate
  - Consoles? Probably not.
    - Thinner drivers
    - Tighter control of the hardware
    - Known configurations
    - No (unpredictable) background tasks
  - PC? Mobile? Definitely!
    - Opaque drivers
    - Compositors
    - Varying configurations
    - Background tasks

Do we really need an API for this?

- Perhaps could determine timing with GPU queries
  - But what about the compositor?
  - Cannot schedule frames later than GPU is done
  - Even if you manage - how to know when to recover?
- Is someone already doing it without the API?
  - The anti-microstutter ~~fix~~ kludge
- API should be available "for the greater good"!
- API is not imposing an approach - heuristics are still up to the developer.
- Not just for games!
  - Video players are in a sad condition today.

Is GOOGLE_display_timing perfect?

- It is incomparably better than the next alternative.
- Slight ambiguity of the “margin”
  - But this only matters in how soon you can recover
  - Not actually smoothness problem, but extra “bonus” performance problem
  - Might be worked on

Praise to the brave engineers who made this possible...

- Dean Sekulić (Croteam)
  - Tracking down the first Sasquatch in the wild (~2012!)
- James Jones (NVIDIA)
  - VDPAU idea
- Karlo Jež (Croteam)
  - Implementing VDPAU prototype and the VK_GOOGLE_display_timing version
- Aaron Leiby (Valve)
  - pointing out problems with our early ideas
- Ian Elliot (Google)
  - defining the VK_GOOGLE_display_timing extension
- Pierre-Loup A. Griffais and Keith Packard (Valve)
  - for the Mesa implementation
- Everyone at Vulkan Advisory Panel
  - long and productive discussions about this
- Andrei Tatarinov and Liam Byrne (Nvidia)
  - for making this talk happen

Thank you!

Questions?

(Wrap up room: Overlook 3022 & 3024)

Alen Ladavac

@AlenL

alenl@croteam.com

The Elusive Frame Timing

A Case for Smoothness Over Speed