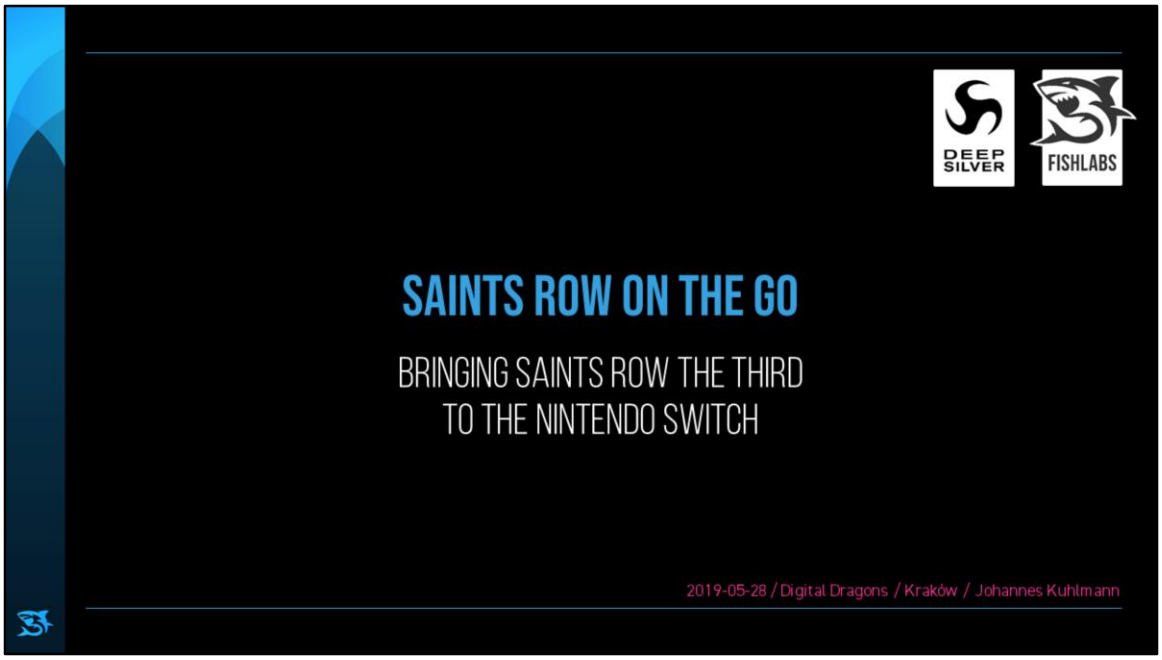

DEEP SILVER
FISHLABS
SAINTS ROW ON THE GO
BRINGING SAINTS ROW THE THIRD
TO THE NINTENDO SWITCH
2019-05-28 / Digital Dragons / Kraków / Johannes Kuhlmann

ABOUT SAINTS ROW THE THIRD…

GAME BY OUR SISTER STUDIO VOLITION
THEY DID AN INCREDIBLE JOB
GAMEPLAY, ASSETS, ENGINE ALL DONE BY THEM
JUST BRINGING IT TO A NEW PLATFORM

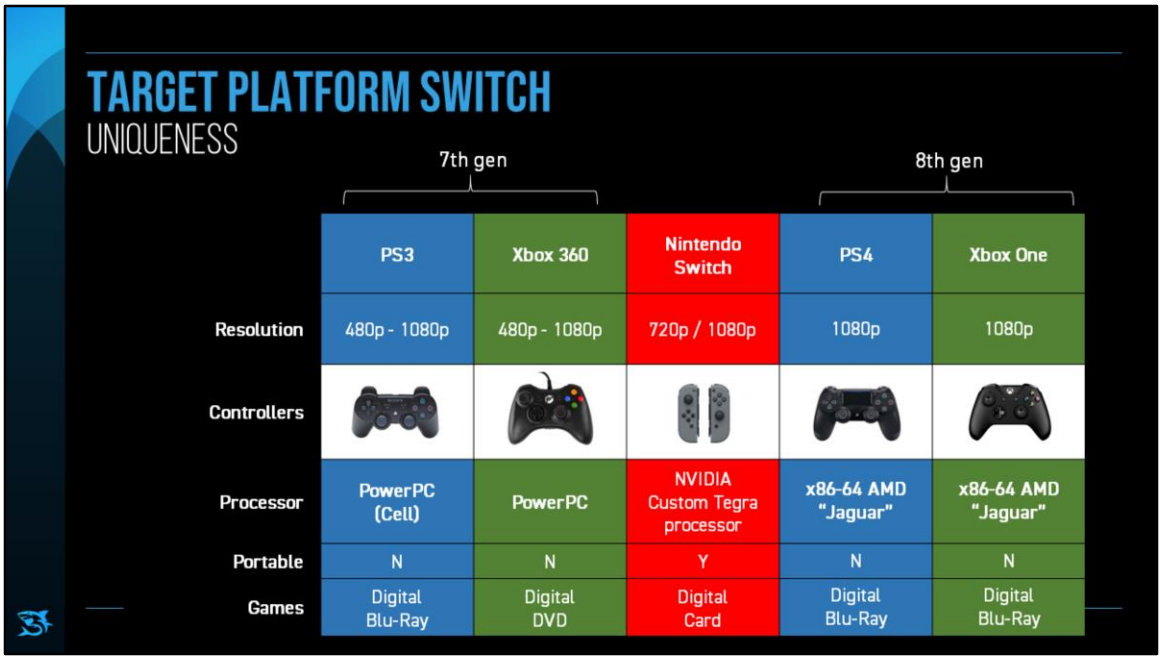
PC/PS3/XBOX 360 TO SWITCH

- WE ONLY DO THE PORTING
- IT'S A CUSTOM ENGINE

THE SWITCH IS REALLY QUITE SUCCESSFUL
MORE THAN 34 MILLION DEVICES SOLD.
MAKES SENSE TO BRING YOUR GAME TO THE SWITCH AS WELL.

--

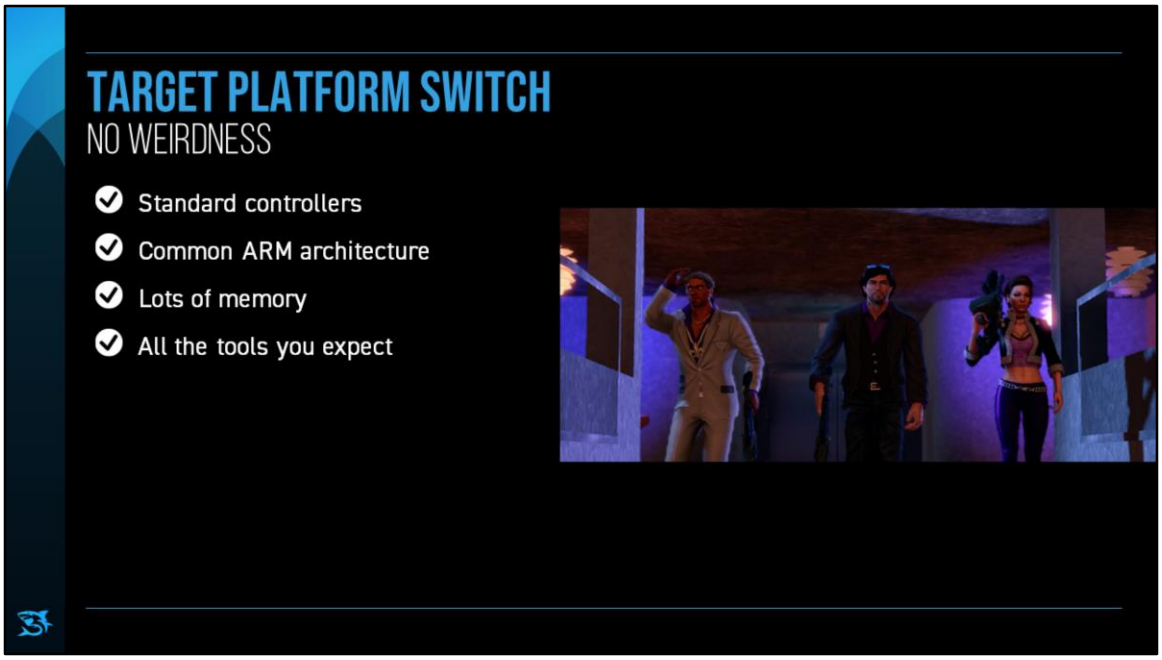
HTTPS://WWW.NINTENDO.CO.JP/IR/EN/FINANCE/HARD_SOFT/



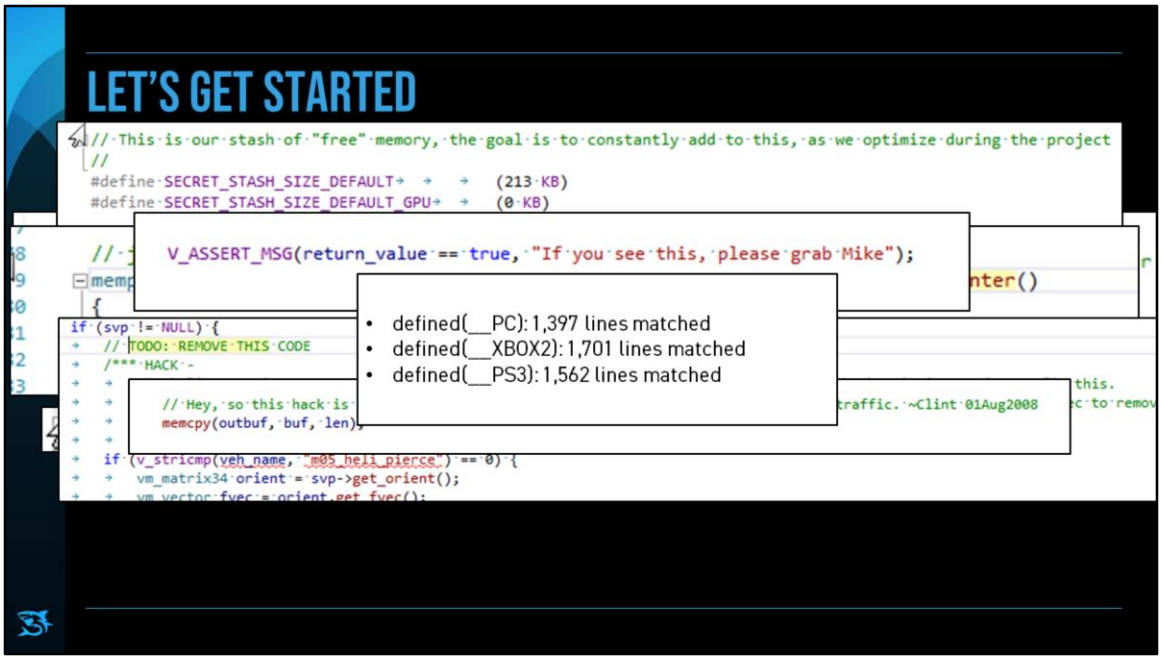
THE SWITCH IS REALLY QUITE IN A UNIQUE SPOT HERE.
NOT LOOKING AT PERFORMANCE HERE, THAT'S MORE DIFFICULT TO QUANTIFY.
SR3 SHIPPED ON GENERATION 7 ORIGINALLY.

--

HTTPS://WWW.NINTENDO.COM/SWITCH/FEATURES/TECH-SPECS/

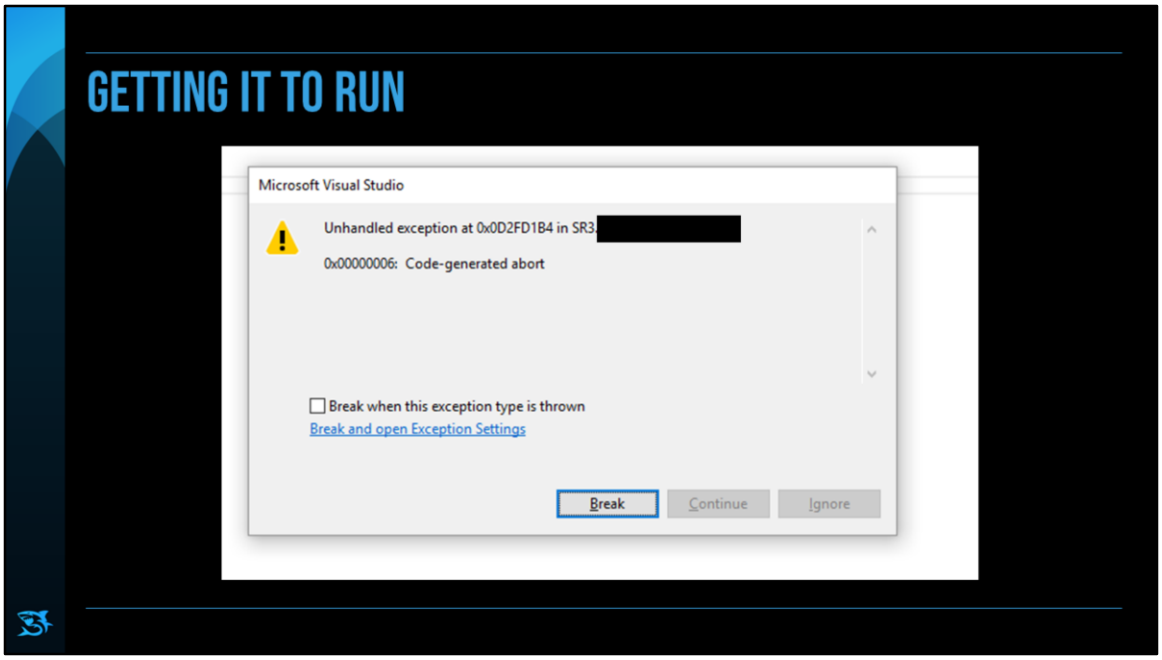


DEVELOPING FOR THE SWITCH IS GREAT.
A DEVICE THAT FULFILLS MOST STANDARDS YOU'RE USED TO ALREADY



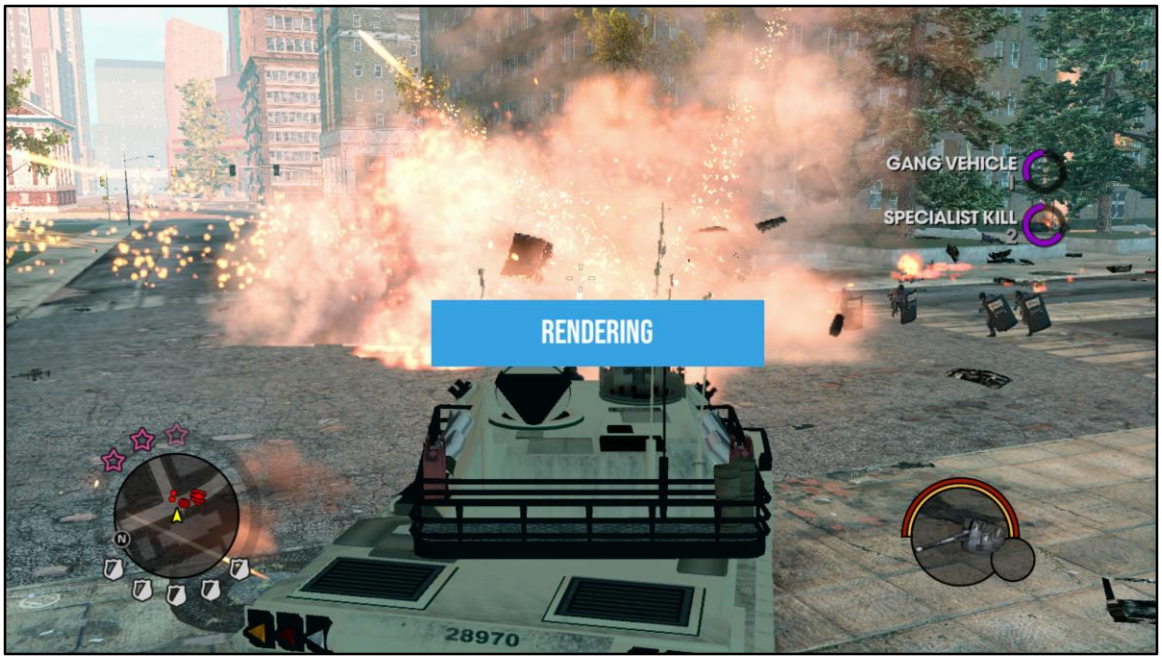
BEFORE WE CAN MAKE A CHANGE WE HAVE TO AT LEAST HAVE A BASIC UNDERSTANDING OF HOW THE GAME WORKS. THIS IS MY FAVORITE PART OF WORKING ON AN EXISTING GAME. FINDING ALL THOSE LITTLE GEMS OF INTERESTING AND FUNNY THINGS HIDDEN THROUGHOUT THE SOURCE CODE.

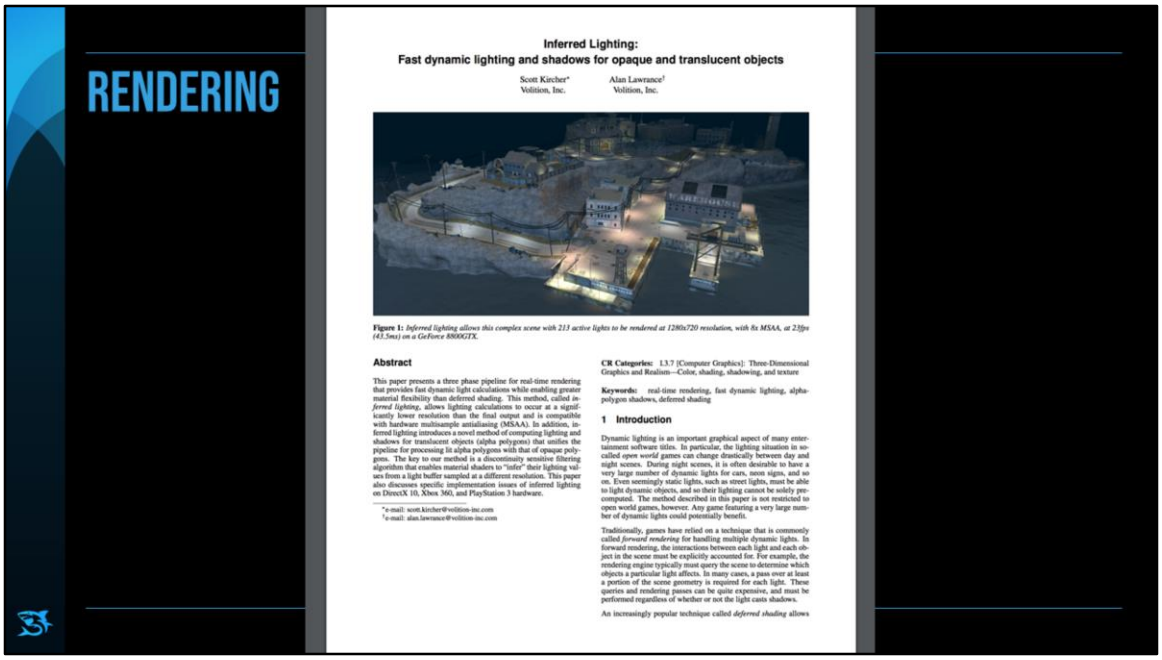
THERE ARE MORE THAN 1000 BLOCKS OF CODE WE'LL HAVE TO ADAPT OR REWRITE FOR THE SWITCH.



ALL THE COMMON TASKS LIKE MEMORY, THREADING, FILE IO, SAVEGAMES, NETWORKING, ETC. HAVE TO BE DONE OF COURSE
TRANSITIONED THE GAME FROM 32 BIT TO 64 BIT.

AFTER GETTING IT TO BUILD, WE HAD A OUR FIRST SUCCESS TO CELEBRATE:
THE FIRST CRASH ON THE NEW PLATFORM. WOOHOO!
THERE WAS OF COURSE STILL A LOT TO DO.


GANG VEHICLE
SPECIALIST KILL
2
RENDERING
28970



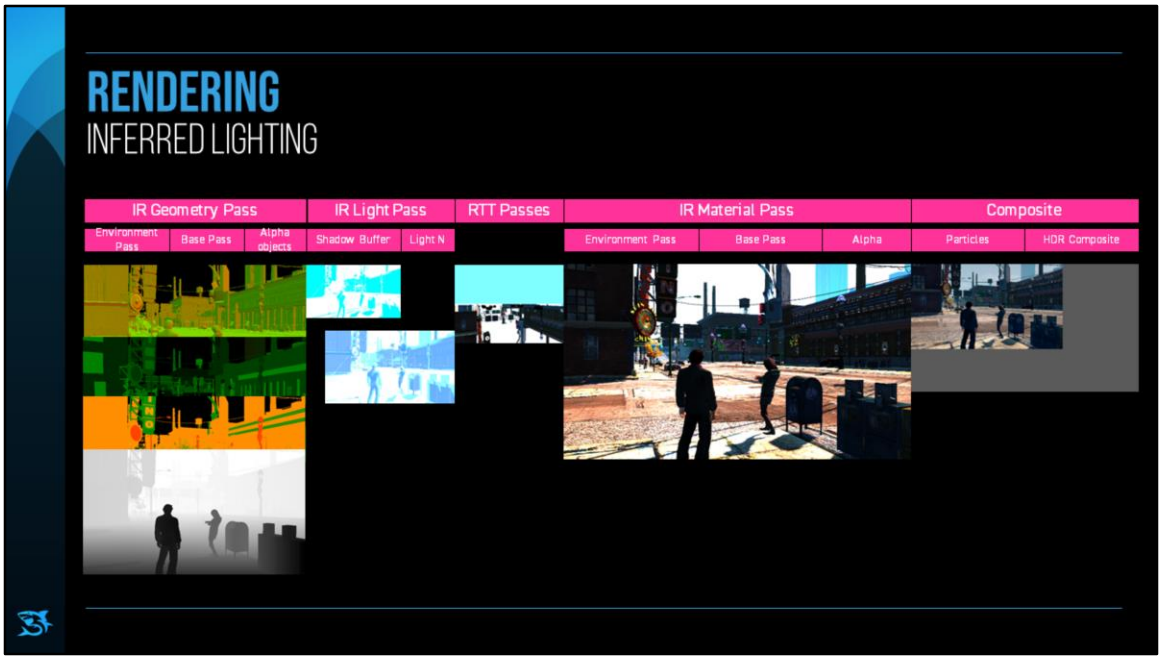
TRIED TO FIND AN OVERLAP IN GRAPHICS APIS BETWEEN THE GAME AND SWITCH, BUT THERE WAS NONE.
SWITCH SUPPORTS: OPENGL, VULKAN, NVN

WE OPTED TO USE THE SWITCH'S OWN GRAPHICS API CALLED NVN.
NVN PROVIDES THE BEST DOCUMENTATION, TOOLS, AND PERFORMANCE.

ONE OF THE BIGGEST TASKS WAS TO GET THE RENDERING RIGHT.
THE RENDERING OF SR3 IS QUITE COMPLEX AND IT'S REALLY USING LOTS OF GRAPHICS API FEATURES.
IT USES THE "INFERRED RENDERING" APPROACH. ONCE WE KNOW THAT, IT WAS EASY TO FIND MORE DOCUMENTATION ABOUT IT ONLINE.

--

HTTP://WWW.STUDENTS.SCIENCE.UU.NL/~3220516/ADVANCEDGRAPHICS/PAPERS/INFERRED_LIGHTING.PDF
HTTPS://WWW.DSVOLITION.COM/PUBLICATIONS/INFERRED-LIGHTING-FAST-DYNAMIC-LIGHTING-AND-SHADOWS-FOR-OPAQUE-AND-TRANSLUCENT-OBJECTS/



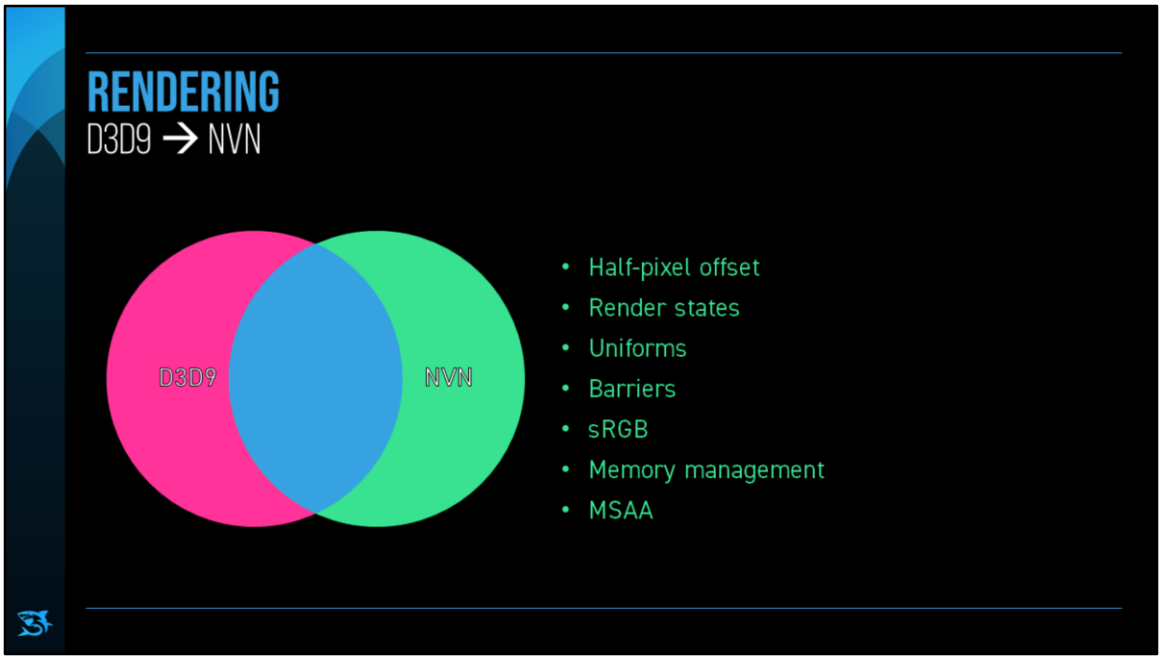
GEOMETRY PASS
- NORMALS
- DSF DATA
- SPEC POWER + LIGHT MASK
- DEPTH

LIGHT PASS
MATERIAL PASS
POST EFFECTS

ADVANTAGES:
MANY NON-SHADOW CASTING LIGHTS
COMPLEX MATERIAL SHADERS
UNIFIED LIGHTING OF TRANSPARENT + OPAQUE OBJECTS
KEEPS MSAA WORKING
-> LIGHTING CAN HAVE DIFFERENT RESOLUTION DUE TO DSF

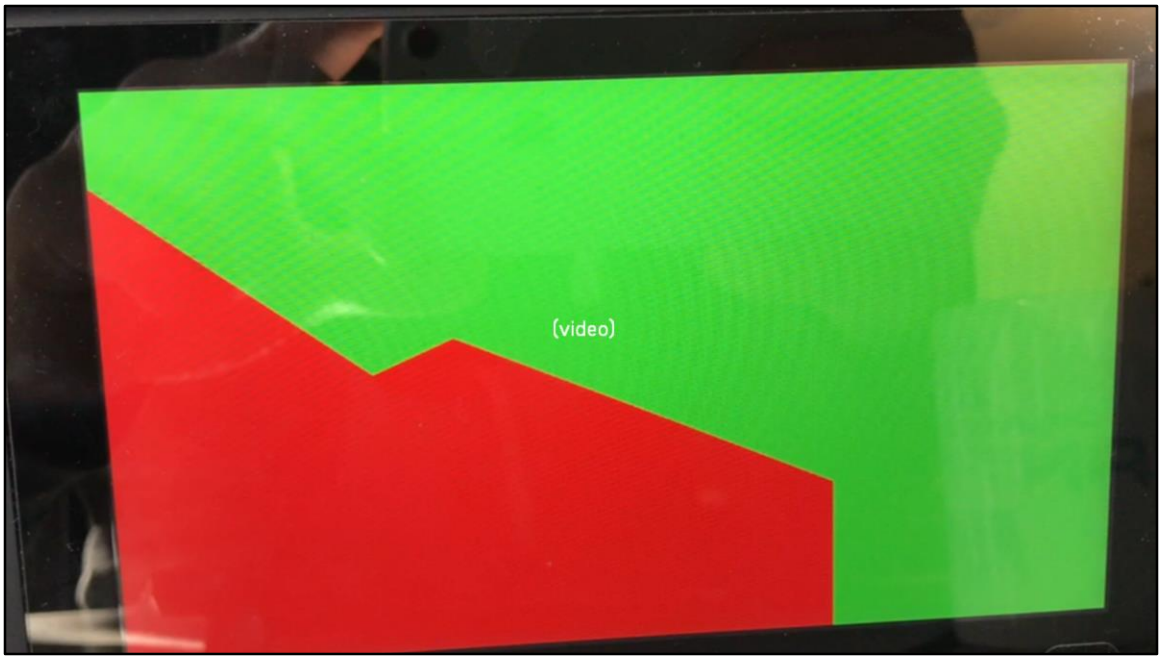
DISADVANTAGES:
HIGHER BASE COST THAN E.G. DEFERRED LIGHTING

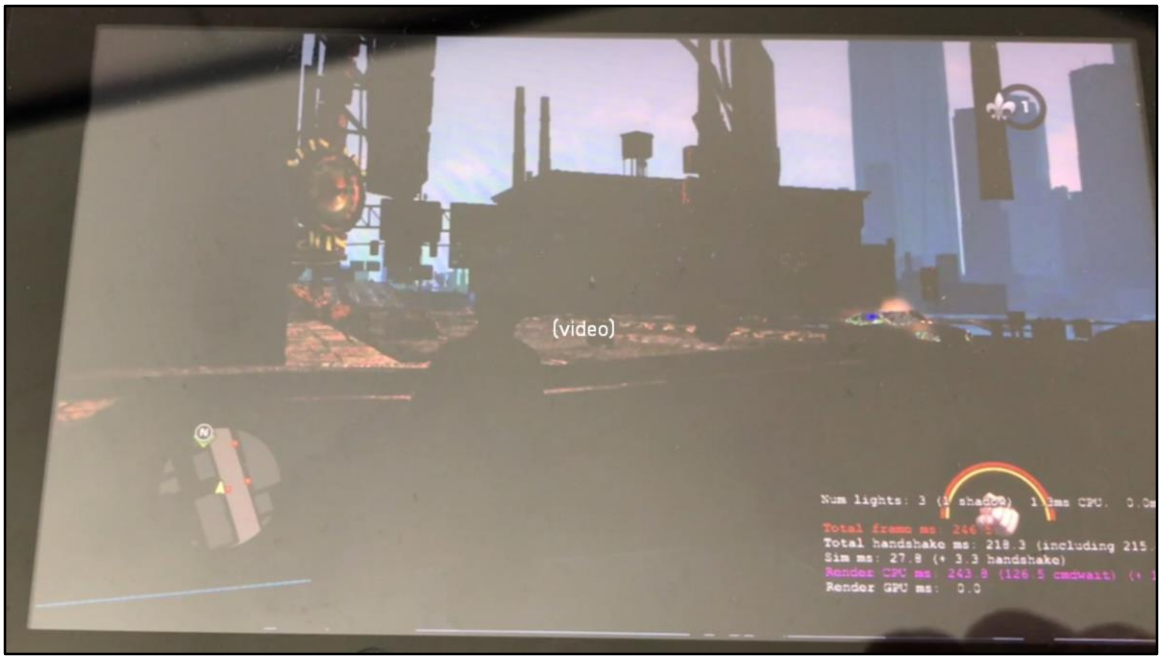
SMALL NUMBER OF LIGHTING LAYERS FOR TRANSPARENT OBJECTS

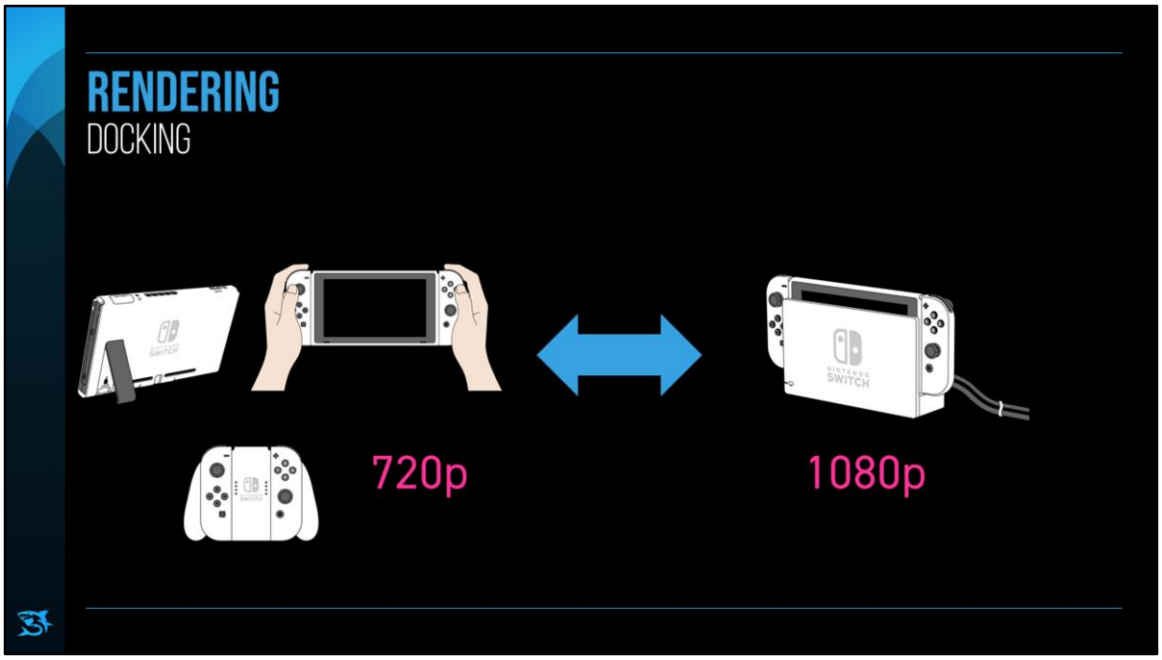


WE BASED THE NEW RENDERING BACKEND ON THE EXISTING DIRECT3D 9 ONE. THIS PROVED TO BE A PRETTY GOOD APPROACH, EVEN THOUGH THERE ARE STILL QUITE A FEW DIFFERENCES.
IT'S RELATIVELY STRAIGHT-FORWARD. IN THE END, ALL GRAPHICS API ARE PRETTY MUCH ALIKE. YOU HAVE SOME RESOURCES, SOME RENDER STATE AND EVENTUALLY DRAW CALLS.

SO THIS IS A LITTLE LIST OF DIFFERENCES THAT YOU HAVE TO WATCH OUT FOR WHEN COMING FROM D3D9.
HALF-PIXEL: SR3 ACCOUNTS FOR THIS A LOT IN THE SHADERS, HAD TO BE DISABLED
RENDER STATES ARE A BIT COARSER THAN D3D9.
UNIFORMS ARE MANAGED IN BUFFERS.
BARRIERS: PRIMITIVE/FRAGMENT ORDERING, UNIFORM DATA, ...
SRGB: TEXTURE VIEWS TO READ AND WRITE IN *_SRGB FORMATS.
MEMORY MANAGEMENT MORE EXPLICIT/MANUAL.
MSAA: EXPLICIT COMMAND BUFFER DOWNSAMPLE OPERATION,


(video)

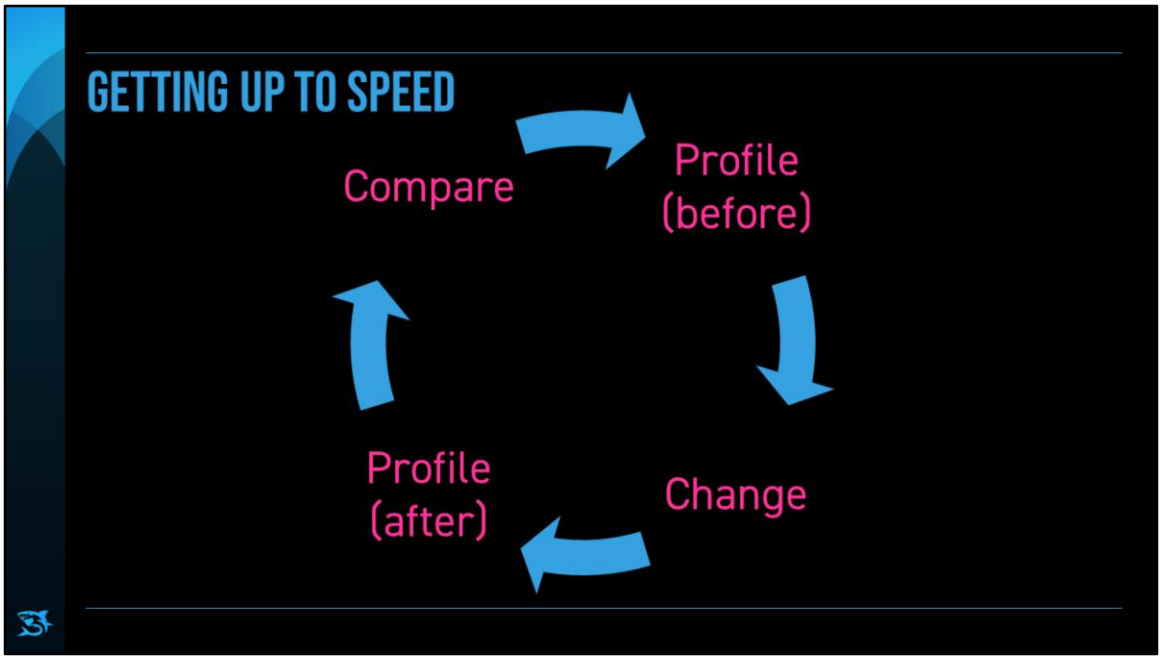

(video)
Total handshake ms: 218.3 (including 215
Sim ms: 27.8 (+ 3.3 handshake)
Render GPU ms: 0.0



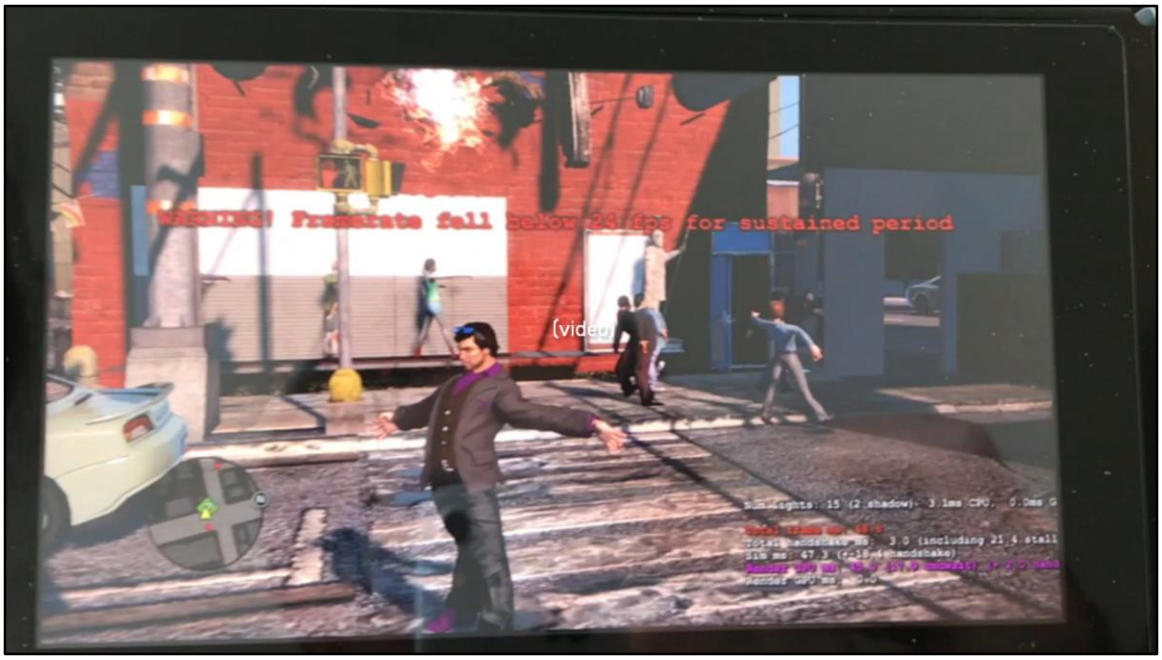
THE SWITCH'S MAIN UNIQUE FEATURE: DOCKING. CAN PLAY IN HANDHELD MODE OR ON THE TV AT HOME.
DOCKED/UNDOCKED PLAY TIME SPLIT 50/50.

HAS TO BE SEAMLESS.
ENGINE/GAME MUST BE PREPARED TO CHANGE RESOLUTION AT RUNTIME
BUT, BECAUSE IT'S ONLY TWO DIFFERENT SPECIFIC RESOLUTIONS YOU CAN ACTUALLY TAKE SOME SHORTCUTS. YOU DON'T HAVE TO COVER THE GENERIC CASE OF JUST ANY RESOLUTION.
WE CREATE ALL RENDER TARGETS ON START-UP IN BOTH SIZES. CREATE THEM IN THE SAME MEMORY ALLOCATIONS TO NOT WASTE ANY MEMORY.

SO, NOW THAT THE GAME IS RUNNING, THE REAL WORK BEGINS.

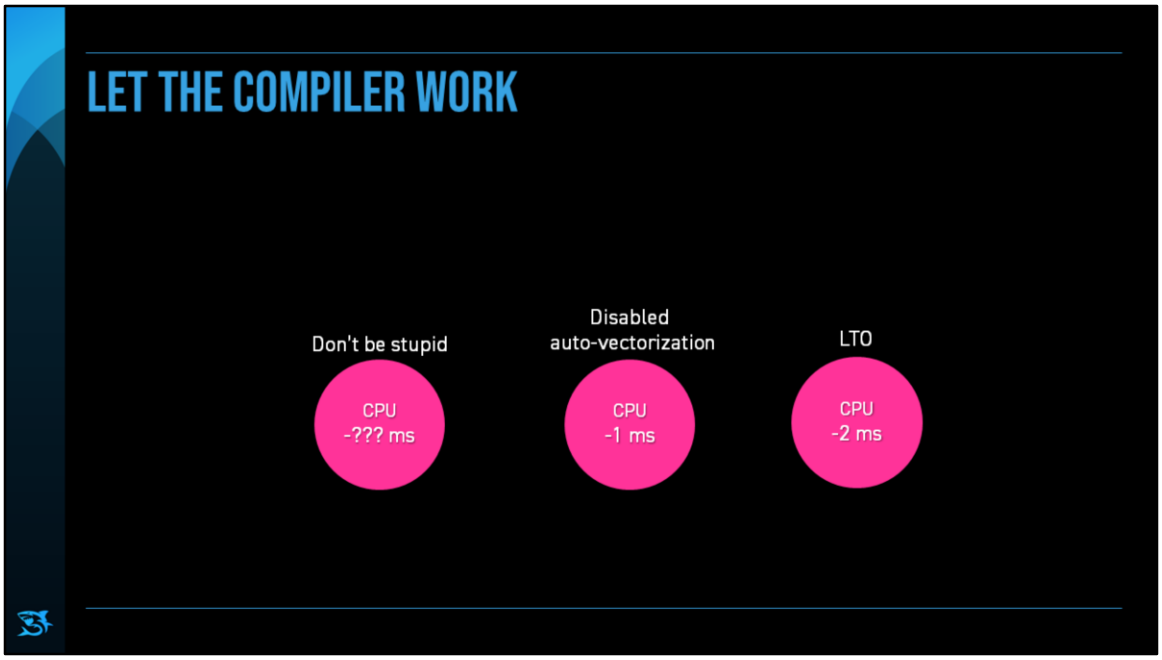


FIND THE BOTTLENECK:
IS IT CPU? IF YES, WHERE?
IS IT GPU? IF YES, WHERE?
NEVER EVER OPTIMIZE ANYTHING WITHOUT MEASURING FIRST.
AND EVEN IF SOME OPTIMIZATION DOESN'T SEEM TO YIELD ANY IMPROVEMENTS, IT MAY BE WORTH TO KEEP IT AROUND – DISABLED OF COURSE. IT STILL TURN OUT BE A BENEFIT LATER OR SIMPLY IN A DIFFERENT SITUATION FROM THE ONE YOU PROFILED BEFORE.



TAKE CARE NOT BREAK STUFF

LET'S LOOK AT SOME CPU OPTIMIZATIONS WE IMPLEMENTED.



DON'T BE STUPID:
ENABLE OPTIMIZATIONS
REMOVE `__attribute__((optnone))`

WHEN YOU'RE PROFILING AND YOU REALLY DRILL DOWN TO THE INSTRUCTION LEVEL IN THE PROFILER AND HE LOADS ARE THE MOST EXPENSIVE INSTRUCTIONS?!
YOU'RE REALLY STARTING TO UNDERSTAND THE IMPACT OF CACHE MISSES ON PERFORMANCE.

THE IDEA IS TO REORDER/REMOVE DATA TO MAKE BETTER USE OF THE CACHE.

ONE BIG OPTIMIZATION WE DID, WAS ADD A LOCAL FOR TEXTURE HANDLES TO THE TEXTURES
INSTEAD OF USING A SHARED GLOBAL CACHE, IT'S NOW SMALL LOCAL CACHE.
SEARCH LINEARLY

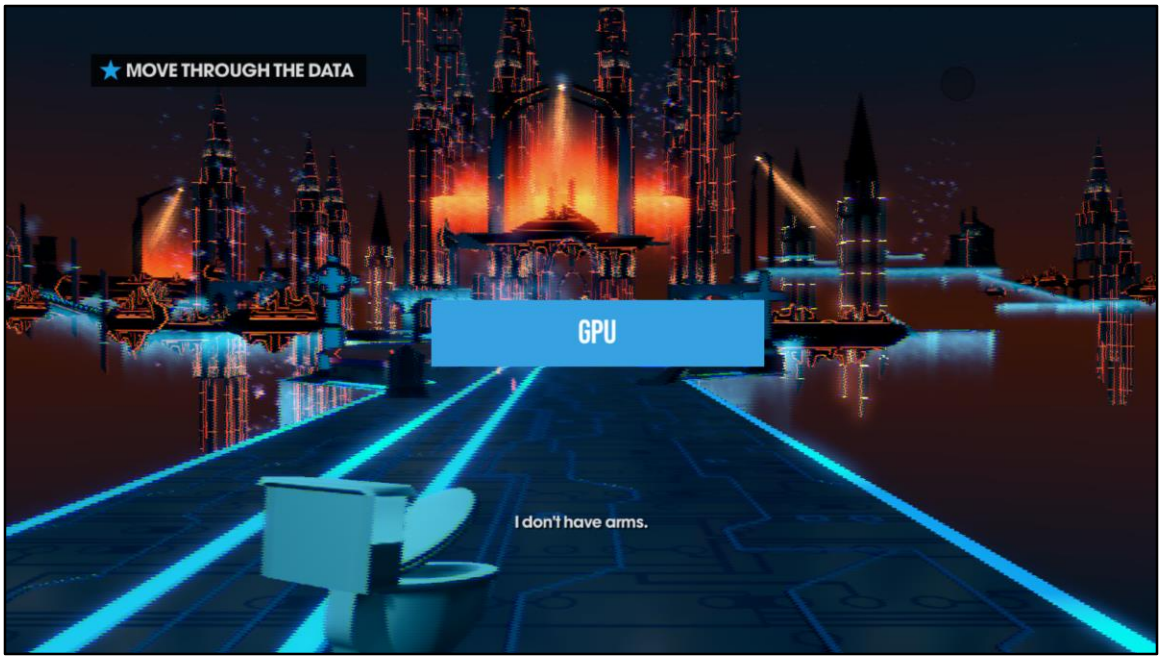
MAKE SURE TAKE ACTUALLY USE LOAD AND STORE FOR SIMD PARAMETERS/RESULTS. DON'T JUST CAST TO FLOAT*.

--

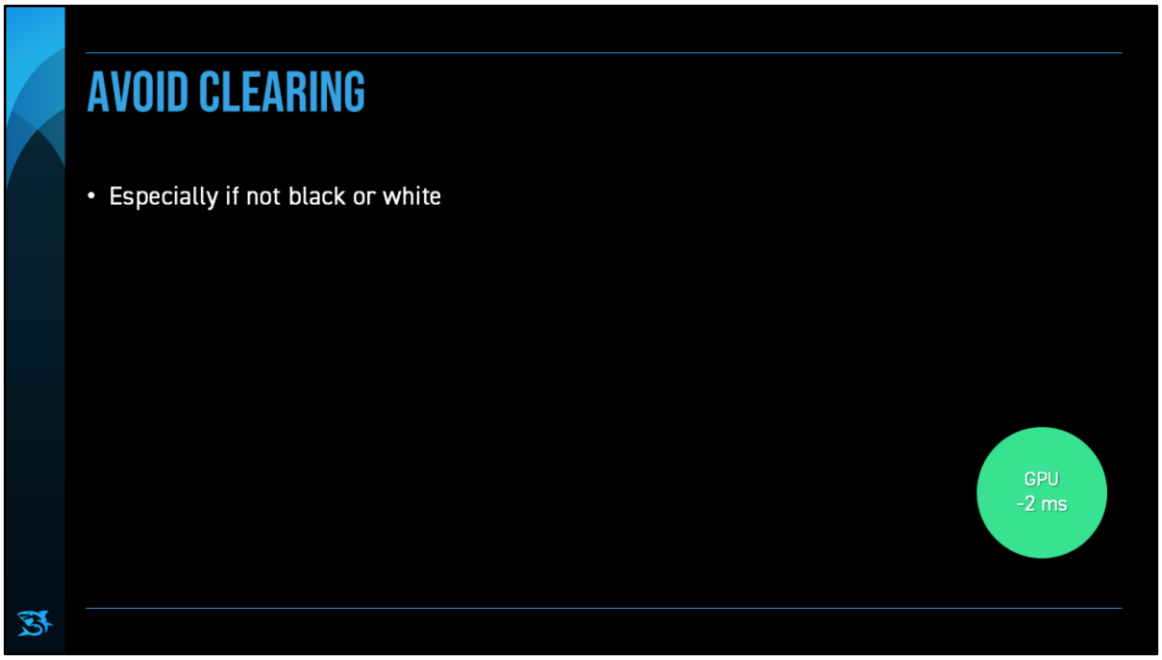
HTTPS://DEVELOPER.ARM.COM/ARCHITECTURES/INSTRUCTION-SETS/SIMD-ISAS/NEON

DON'T COPY IF POSSIBLE.
MEMCPY INSTEAD OF ASSIGNMENTS. ONLY FOR POD TYPES.
INSTEAD OF INDIVIDUAL MEMCPYS, DO A BIG MEMCPY.

SIMD CAN BE FASTER THAN MEMCPY.
REQUIRES ALIGNED START ADDRESSES AS WELL AS SIZES
THIS MAY BE BENEFICIAL IN SOME CASES, BUT NOT IN OTHERS.
SHOULD BE PROFILED ON A PER-CASE BASIS.
WE WERE ABLE TO SPEED UP SOME SMALLER COPIES THIS WAY.

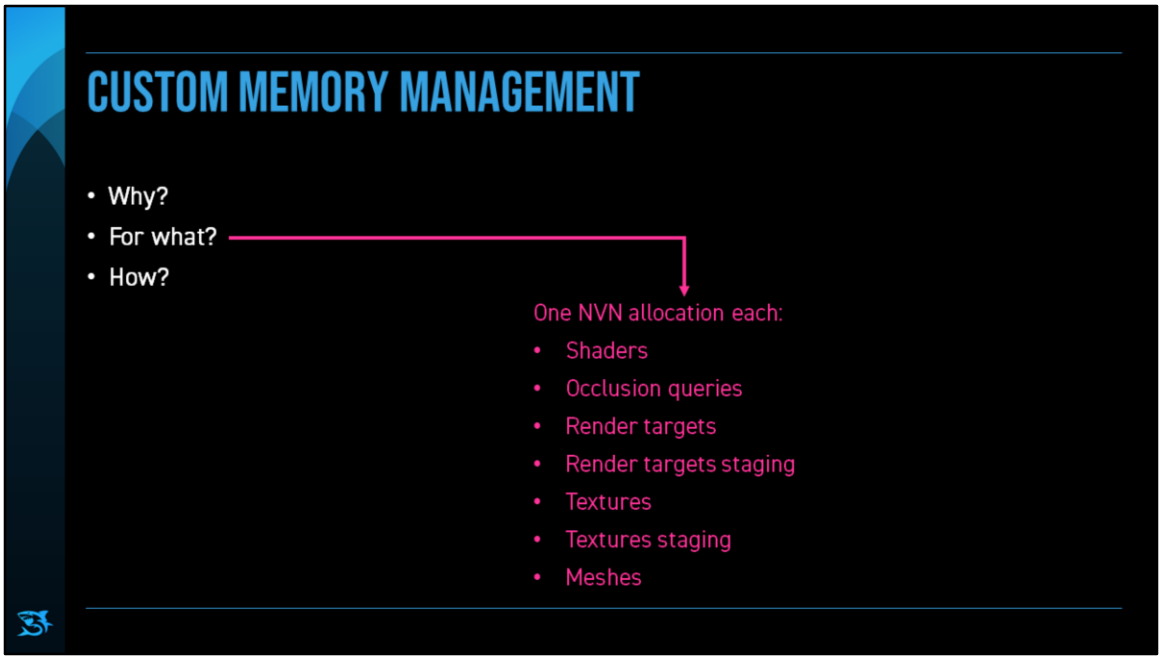


LET'S LOOK AT SOME GPU OPTIMIZATIONS WE IMPLEMENTED.



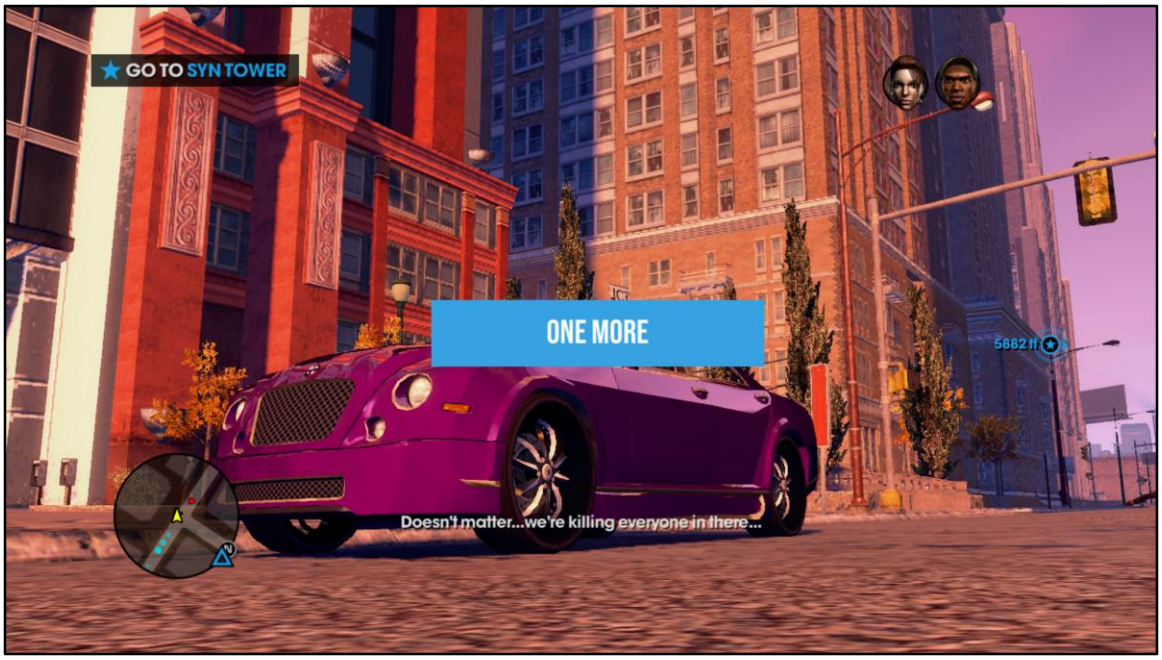
IF YOU RENDER OF THE COMPLETE RENDER TARGET ANYWAY, NO NEED TO CLEAR.
MATERIAL RT WAS BEING CLEARED TO SOME SHADE OF BLUE WHICH WAS RELATIVELY EXPENSIVE.

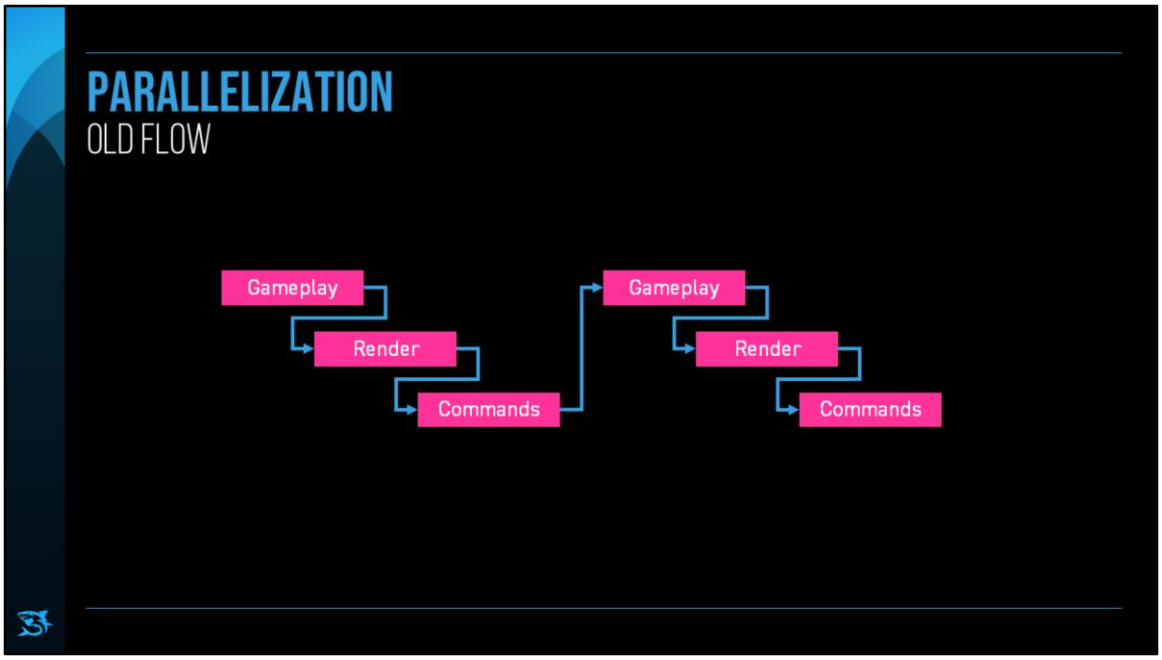
RESOLUTION OF GEOMETRY BUFFERS DOESN'T SCALE WITH THE DISPLAY RESOLUTION WHICH GIVES US A PRETTY GOOD PERFORMANCE IMPROVEMENT.



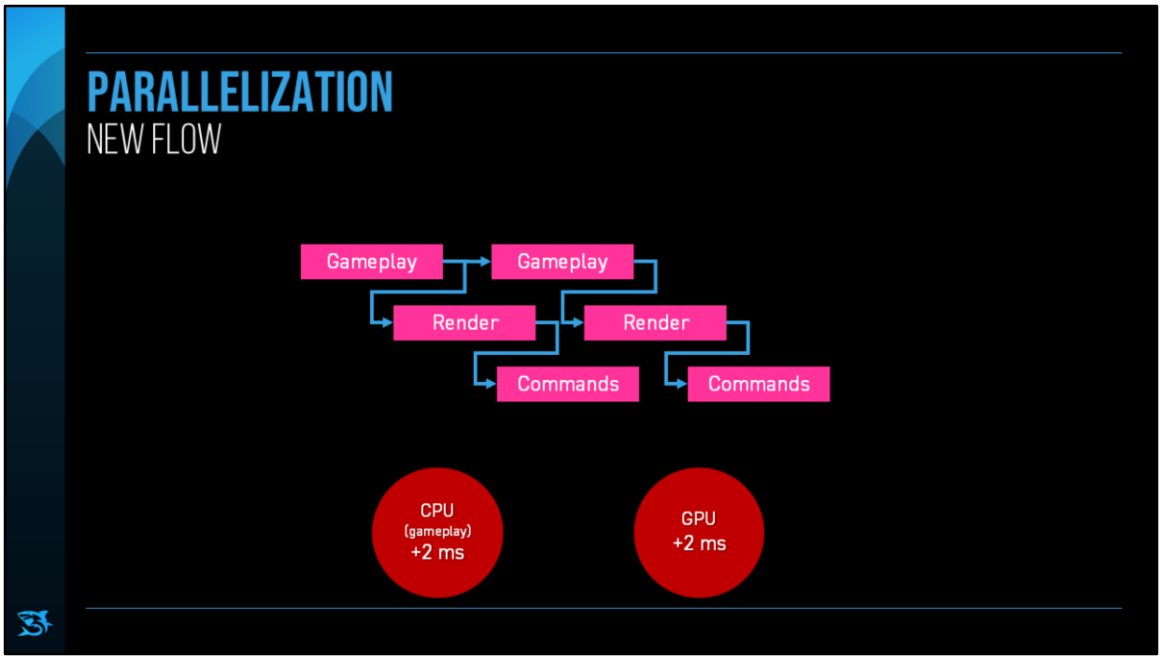
WHY?
PERFORMANCE (ALLOCATIONS ARE EXPENSIVE), KEEPING EYE ON BUDGETS, LIMITS ON ALLOCATIONS (USE FEWER NVN ALLOCATIONS).

HOW?
ONE BIG ALLOCATION + REUSE EXISTING PAGE ALLOCATOR FROM SR3.
INDIVIDUAL ASSETS ARE THEN IDENTIFIED USING OFFSETS INTO THESE ALLOCATIONS.


GO TO SYN TOWER
ONE MORE
5882 ft
Doesn't matter...we're killing everyone in there...



NEXT GAMEPLAY WAS NOT BEING TRIGGERED BEFORE PREVIOUS FRAME COMPLETELY DONE. GAMEPLAY/RENDER/COMMANDS RUN ON DIFFERENT THREADS AND THERE IS SOME OVERLAP, BUT STILL SOME TIME WASTED.



SO TRIGGER NEXT GAMEPLAY FRAME IMMEDIATELY AFTER PREVIOUS IS DONE. MAKES BETTER USE OF AVAILABLE CPU TIME.

HOWEVER, THE GAMEPLAY CODE ASSUMES OCCLUSION QUERY RESULTS FROM THE PREVIOUS FRAME ARE ALREADY AVAILABLE WHICH THEY ARE NOT ANYMORE NOW.

DISABLED OCCLUSION QUERIES WHICH LEADS TO WORSE PERFORMANCE ON CPU AND GPU. BUT, BECAUSE WE START EARLIER, IT'S ACTUALLY A PERFORMANCE IMPROVEMENT.

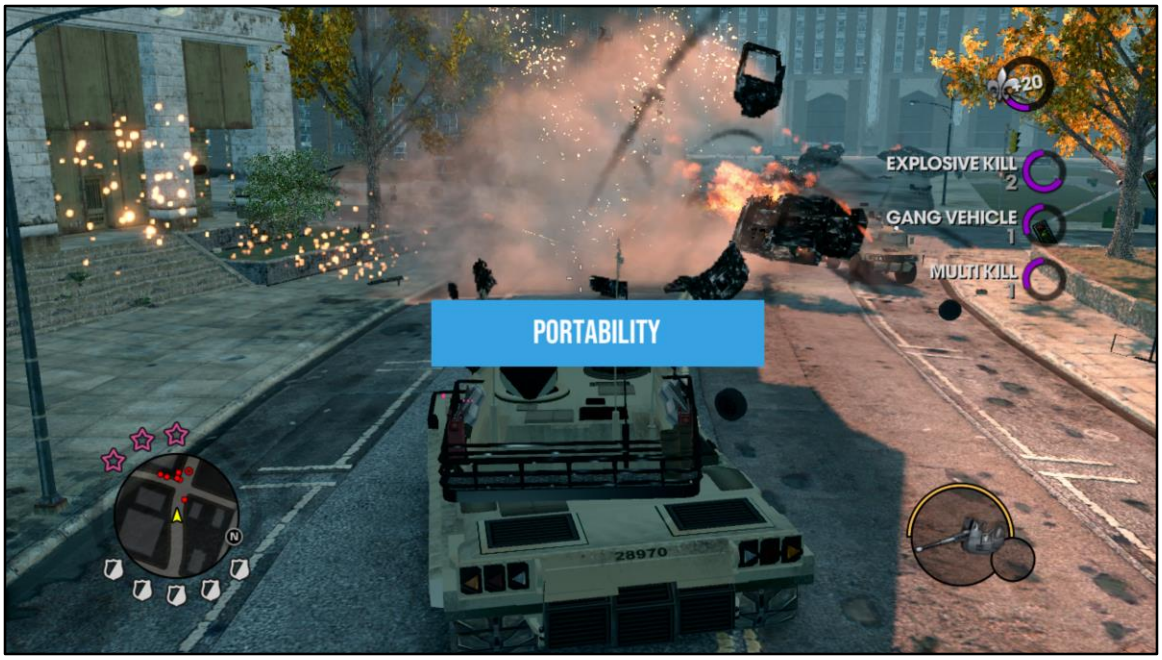
ALSO NEED TO BE CAREFUL WITH GAMEPLAY GENERATED DATA THAT IS TO BE CONSUMED BY THE GPU.

ANOTHER THING WE HAD TO DO WAS TO OPTIMIZE FOR SIZE ON DISK.
THE GAME WAS SUPPOSED TO FIT ON AN 8 GB GAME CARD WITH ALL ITS DLCS.

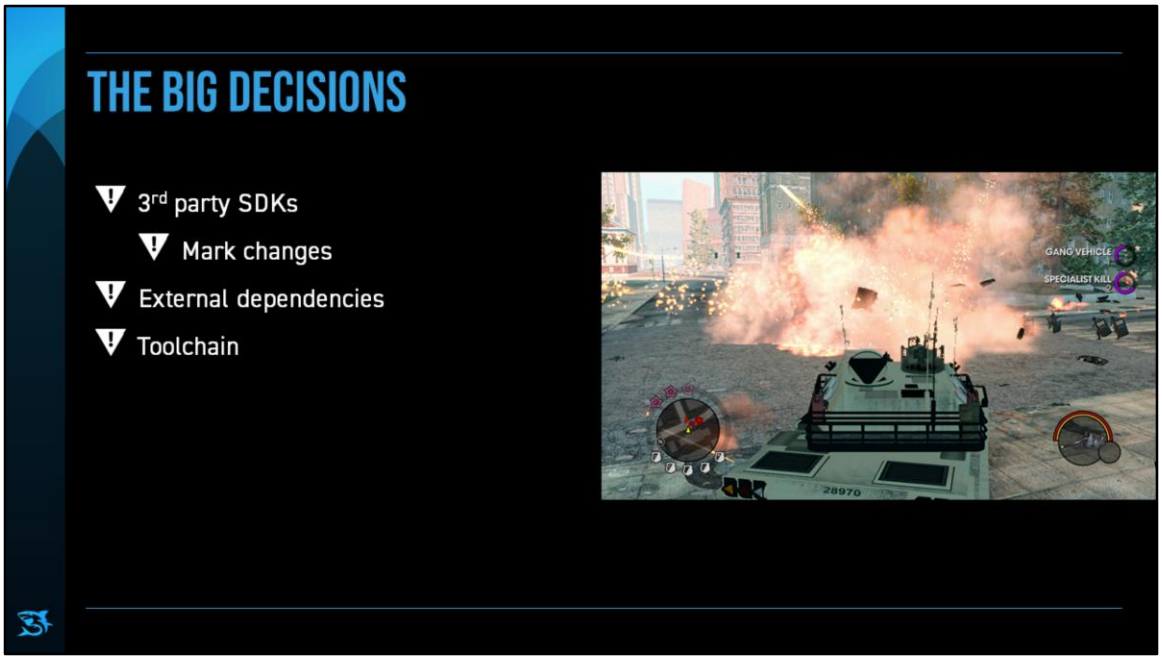
THE SETUP WAS THAT "STREAMING CHUNKS" WOULD INCLUDE THE TEXTURES WHICH WERE THUS REPLICATED MULTIPLE TIMES ON DISK IN ORDER TO REDUCE SEEK TIMES ON SPINNING DISKS.
THE SOLUTION WAS TO HAVE ALL TEXTURES SEPARATE AND LOAD THEM WHEN NEEDED. THIS WAS POSSIBLE BECAUSE THE SWITCH (WITH ITS SOLID STATE DISK) HAS BETTER SEEK TIMES THAN OPTICAL DISKS.

AFTER THAT, WE WERE IN THE SPACE CONSTRAINTS BY JUST A FEW MB.

LATER, WE FOUND OUT THAT WE HAD ACCIDENTALLY MISSED SOME AUDIO-RELATED DATA, MORE SPECIFICALLY LIP-SYNC AND SUBTITLES FOR ANY DIALOG SPOKEN OUTSIDE OF CUTSCENES. WE HAD TO PATCH THAT IN.
HAD WE ACTUALLY DONE THIS CORRECTLY IN THE FIRST PLACE, THE GAME WOULD NOT HAVE FIT.



HOW CAN YOU MAKE SURE YOUR PROJECT IS GOING TO BE EASILY PORTABLE TO OTHER PLATFORMS?
IF YOUR GAME IS SUCCESSFUL THERE WILL BE DEMAND FOR YOUR GAME TO BE ON ANY PLATFORM IMAGINABLE.
OF COURSE YOU CAN'T FORESEE ALL THESE PLATFORMS AND YOU CAN'T ADD SUPPORT JUST YET.
THERE ARE SOME EASY GUIDELINES HERE TO GO BY THAT HELP YOU TO NOT PROGRAM YOURSELF INTO A DEAD END AND THAT MAKE EASIER TO ADD A PLATFORMS TO YOUR CODEBASE.



CHOOSE YOUR 3RD PARTY DEPENDENCIES (SDKS, ENGINES) WISELY. YOU SHOULD HAVE TRUST IN THEM TO BE AROUND IN A COUPLE OF YEARS AND SUPPORT THOSE PLATFORMS YOU WANT TO SUPPORT.

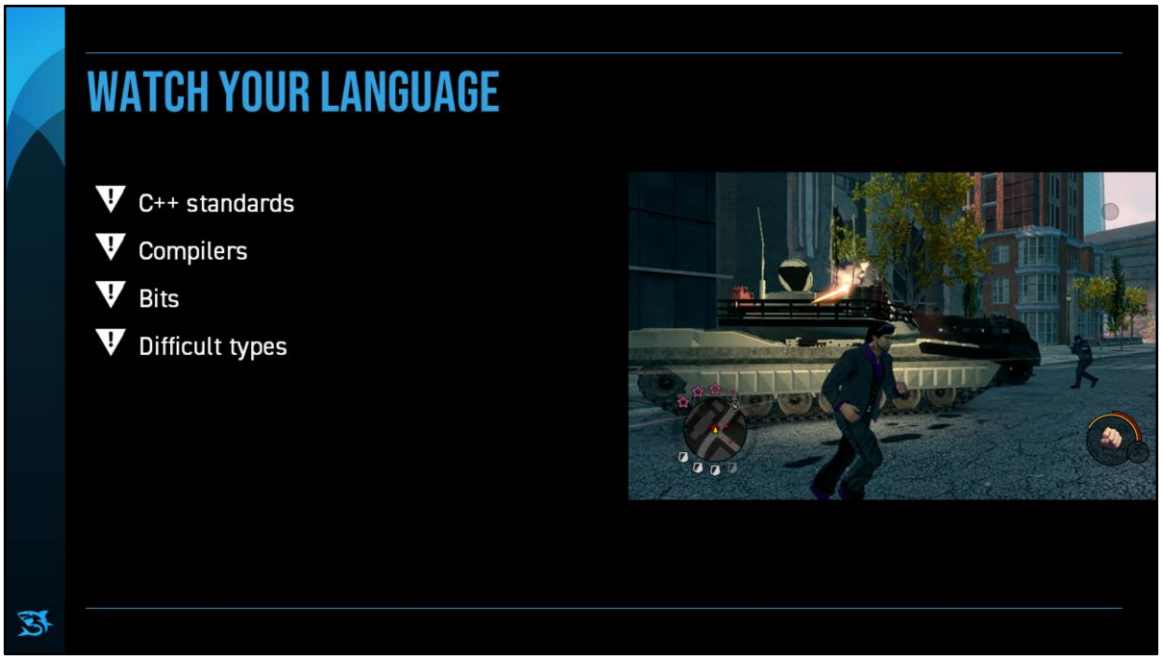
AT LEAST, GET THE SOURCE CODE SO THAT YOU ARE ABLE TO PORT OVER THE 3RD PARTY CODE OVER TO NEW PLATFORMS YOURSELF IF THE VENDOR IS NOT DOING IT.

IF YOU HAVE THE SOURCE AND MAKE CHANGES, MARK THEM CLEARLY SO THAT YOU CAN RE-APPLY THEM ONCE YOU WANT TO UPGRADE THE 3RD PARTY CODE.

ANYTHING EXTERNAL

- SHOULD BE KEPT CLOSE / INCLUDED IN REPO
- WILL ALSO HAVE TO WORK IN DEV ENVIRONMENT FOR NEW PLATFORMS

1. Abstract away those systems which traditionally must be adapted/rewritten. Think input, file IO, savegames, networking, …
2. Stick to one searchable thing, e.g. #ifdef PLATFORM_XYZ
3. Provide fallback/reference implementations if possible.
4. If not possible, make compilation fail if platform support is missing (#else #error platform not supported).



LOOKING AT THE ACTUAL CODE YOU WRITE.

1. CHOOSE A C++ STANDARD AND STICK TO IT. THE OLDER THE STANDARD THE MORE LIKELY IT'S SUPPORTED ON MORE PLATFORMS.
2. SHOULD WORK ON CURRENT COMPILERS (CLANG, MSVC, MAYBE GCC).
3. USE TYPES THAT AVOID BITNESS WHEN DEALING WITH SIZES AND POINTERS -> USE UINTPTR_T, SIZE_T
4. AVOID DIFFICULT TYPES, E.G. WCHAR_T CAN VARY IN SIZE.

SUMMARY

1. Switch is a great target platform.
2. Some optimization may be required.
3. Porting difficulty depends on preconditions.

THANK YOU!

Questions?

Contact: j.kuhlmann@dsfishlabs.com
@j66k

Website: https://dsfishlabs.com/