



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/](https://www.nintendo.co.jp/ir/en/finance/hard_soft/)

TARGET PLATFORM SWITCH

UNIQUENESS

	7th gen			8th gen	
	PS3	Xbox 360	Nintendo Switch	PS4	Xbox One
Resolution	480p - 1080p	480p - 1080p	720p / 1080p	1080p	1080p
Controllers					
Processor	PowerPC (Cell)	PowerPC	NVIDIA Custom Tegra processor	x86-64 AMD "Jaguar"	x86-64 AMD "Jaguar"
Portable	N	N	Y	N	N
Games	Digital Blu-Ray	Digital DVD	Digital Card	Digital Blu-Ray	Digital Blu-Ray

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/](https://www.nintendo.com/switch/features/tech-specs/)

TARGET PLATFORM SWITCH

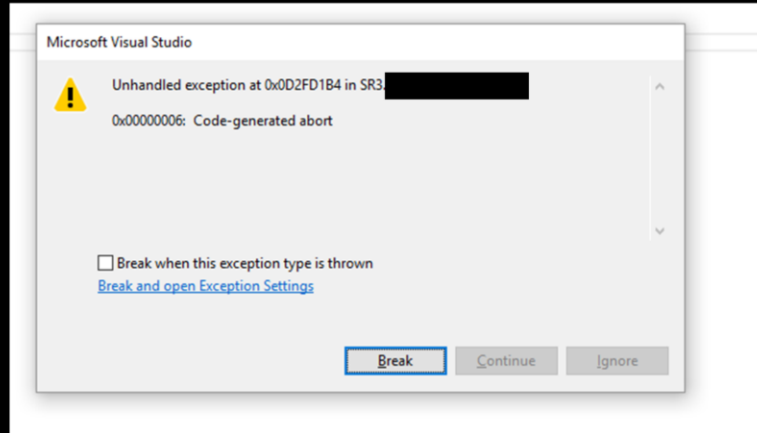
NO WEIRDNESS

- ✓ Standard controllers
- ✓ Common ARM architecture
- ✓ Lots of memory
- ✓ All the tools you expect



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

GETTING IT TO RUN



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. WOHHOO!

THERE WAS OF COURSE STILL A LOT TO DO.



RENDERING

Inferred Lighting: Fast dynamic lighting and shadows for opaque and translucent objects

Scott Kiecher*
Volition, Inc.

Alan Lawrence†
Volition, Inc.



Figure 1: Inferred lighting allows this complex scene with 213 active lights to be rendered at 1280x720 resolution, with 8x MSAA, at 23fps (4:3) on a GeForce 8800GTX.

Abstract

This paper presents a three phase pipeline for real-time rendering that provides fast dynamic light calculations while enabling greater material flexibility than deferred shading. This method, called *inferred lighting*, allows lighting calculations to occur at a significantly lower resolution than the final output and is compatible with hardware multisample antialiasing (MSAA). In addition, inferred lighting introduces a novel method of composing lighting and shadows for translucent objects (alpha polygons) that unifies the pipeline for processing both opaque polygons with that of opaque polygons. The key to our method is a discontinuity sensitive filtering algorithm that enables translucent shadows to “hide” their lighting values from a light buffer sampled at a different resolution. This paper also discusses specific implementation issues of inferred lighting on DirectX 10, Xbox 360, and PlayStation 3 hardware.

*e-mail: scott.kiecher@volition-inc.com

†e-mail: alan.lawrence@volition-inc.com

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture.

Keywords: real-time rendering, fast dynamic lighting, alpha-polygon shadows, deferred shading.

1 Introduction

Dynamic lighting is an important graphical aspect of many entertainment software titles. In particular, the lighting situation in so-called open world games can change drastically between day and night scenes. During night scenes, it is often desirable to have a very large number of dynamic lights for cars, neon signs, and so on. Even seemingly static lights, such as street lights, must be able to light dynamic objects, and so their lighting cannot be solely pre-computed. The method described in this paper is not restricted to open world games, however. Any game featuring a very large number of dynamic lights could potentially benefit.

Traditionally, games have relied on a technique that is commonly called *forward rendering* for handling multiple dynamic lights. In forward rendering, the intersections between each light and each object in the scene must be explicitly accounted for. For example, the rendering engine typically must query the scene to determine which objects a particular light affects. In many cases, a pass over at least a portion of the scene geometry is required for each light. These queries and rendering passes can be quite expensive, and must be performed regardless of whether or not the light casts shadows. An increasingly popular technique called *deferred shading* allows

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](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/](https://www.dsvolition.com/publications/inferred-lighting-fast-dynamic-lighting-and-shadows-for-opaque-and-translucent-objects/)

RENDERING

INFERRED LIGHTING



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

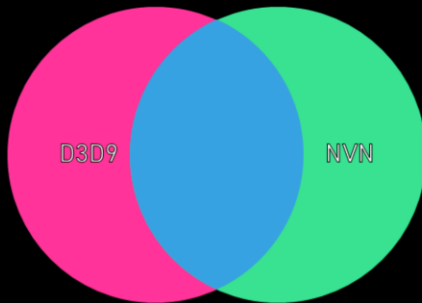
DISADVANTAGES:

HIGHER BASE COST THAN E.G. DEFERRED LIGHTING

SMALL NUMBER OF LIGHTING LAYERS FOR TRANSPARENT OBJECTS

RENDERING

D3D9 → NVN



- Half-pixel offset
- Render states
- Uniforms
- Barriers
- sRGB
- Memory management
- MSA

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.

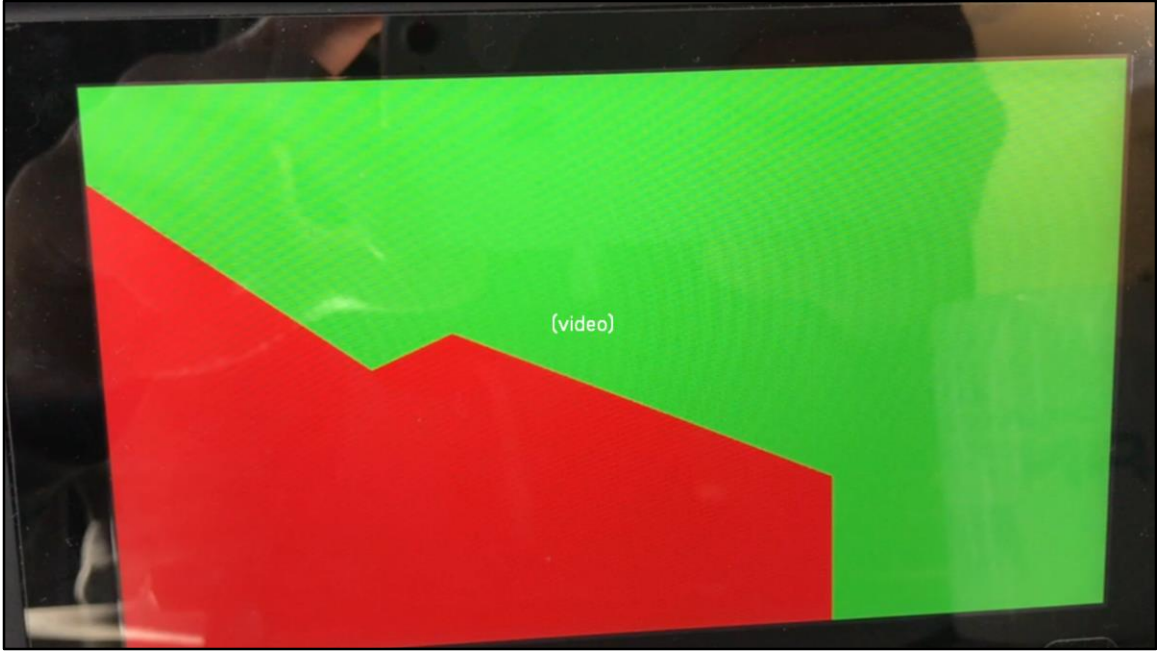
UNIFORMS ARE MANAGED IN BUFFERS.

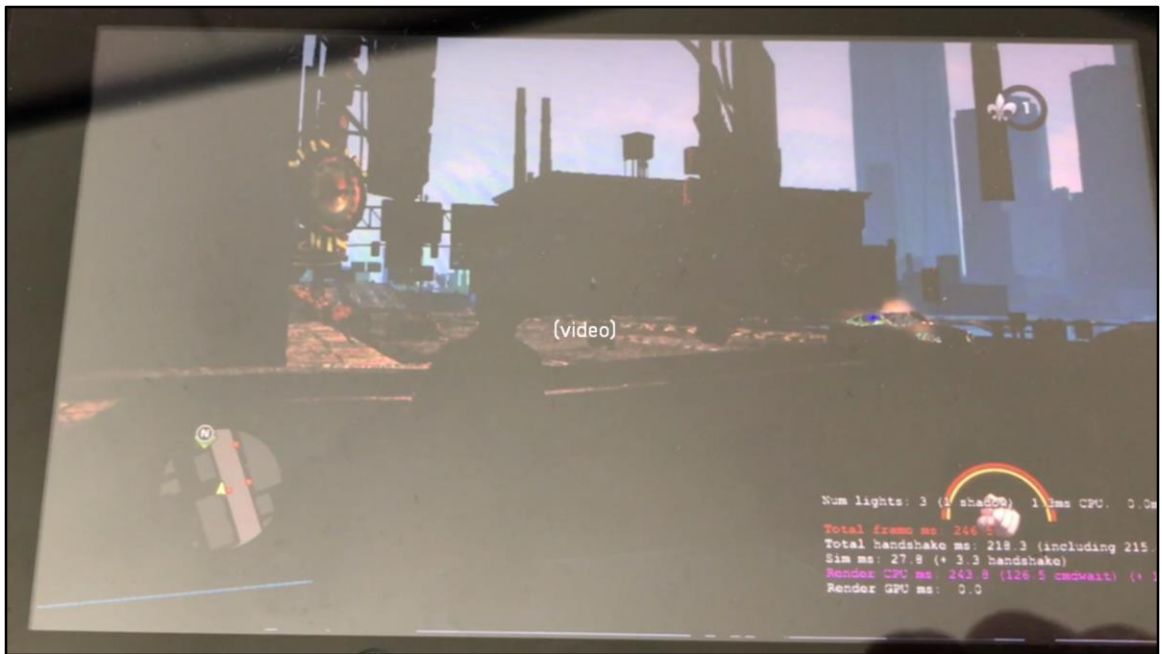
BARRIERS: PRIMITIVE/FRAGMENT ORDERING, UNIFORM DATA, ...

SRGB: TEXTURE VIEWS TO READ AND WRITE IN *_SRGB FORMATS.

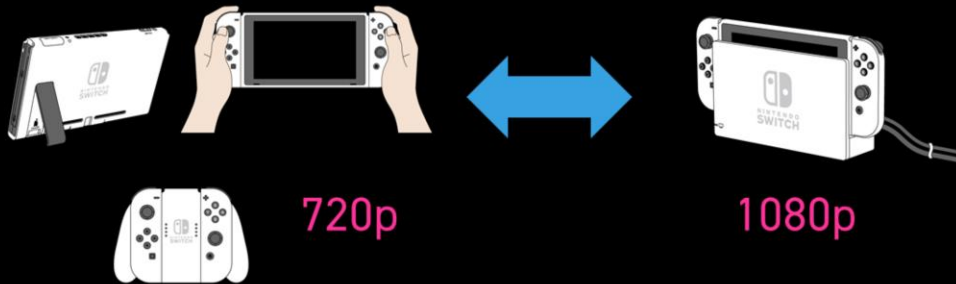
MEMORY MANAGEMENT MORE EXPLICIT/MANUAL.

MSAA: EXPLICIT COMMAND BUFFER DOWNSAMPLE OPERATION,





RENDERING DOCKING



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

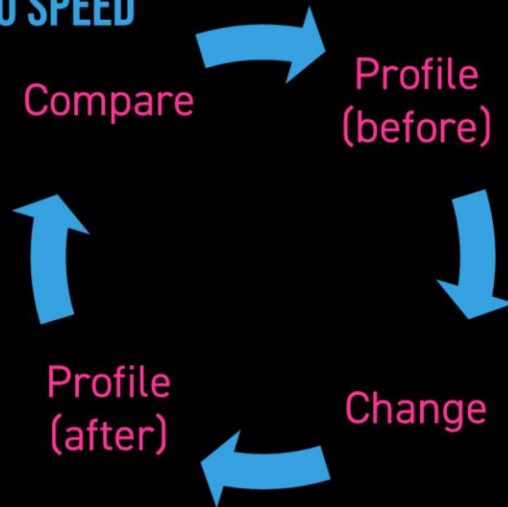
BUT, BECAUSE IT'S ONLY TWO DIFFERENT SPECIFIC RESOLUTIONS YOU CAN ACTUALLY TAKE SOME SHORTCUTS. YOU DON'T HAVE TO COVER THE GENERIC CASE OF JUST ANY RESOLUTION.

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.

GETTING UP TO SPEED



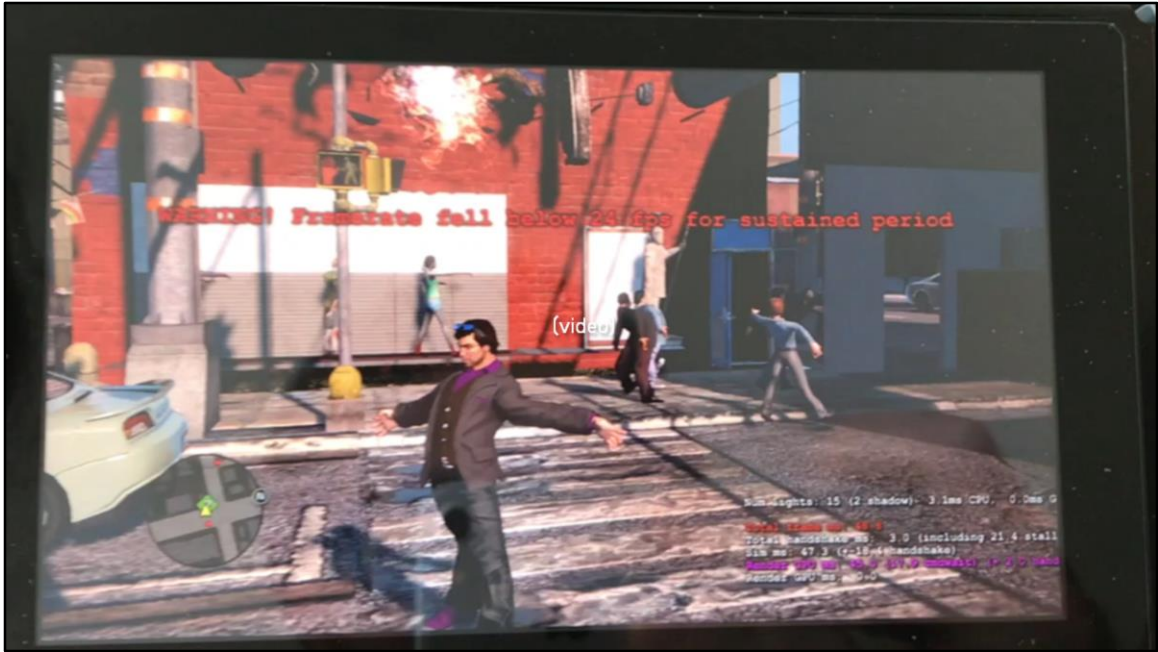
FIND THE BOTTLENECK:

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.

LET THE COMPILER WORK

Don't be stupid



Disabled
auto-vectorization



LTO



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

DATA-ORIENTED DESIGN

```
.L27:
0x009AA0B1C    40    0 9B004822    MADD    x2, x1, x0, x18
type_mat3x3.inl:597 m[0][0] * v.x + m[1][0] * v.y + m[2][0] * v.z,
0x009AA0B20    168   0 2D444526    LDP     s6, s17, [x9, #32]
type_vec3.inl:110 x(s0),
0x009AA0B24    364   0 2D404047    LDP     s7, s16, [x2]
type_mat3x3.inl:598 m[0][1] * v.x + m[1][1] * v.y + m[2][1] * v.z,
0x009AA0B28    0     0 1E3108F1    FMUL   s17, s7, s17
type_mat3x3.inl:599 m[0][2] * v.x + m[1][2] * v.y + m[2][2] * v.z;
0x009AA0B2C    22    0 2D454D32    LDP     s18, s19, [x9, #40]
type_mat3x3.inl:597 m[0][0] * v.x + m[1][0] * v.y + m[2][0] * v.z,
0x009AA0B30    0     0 1E2608E6    FMUL   s6, s7, s6
0x009AA0B34    0     0 1E330A13    FMUL   s19, s16, s19
```

CPU
-0.5 ms

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

SIMD

- SSE → ARM NEON
- Loads & stores

```
// Some examples
void vecf4_store4(float* ptr, VECF4_ARG value)
{
    vst1q_f32(ptr, value);
}

vecf4 vecf4_load4(const float* values)
{
    return vld1q_f32(values);
}

vecf4 vecf4_add(VECF4_ARG v0, VECF4_ARG v1)
{
    return vaddq_f32(v0, v1);
}

bool vecf4_is_near_equal4(VECF4_ARG v0, VECF4_ARG v1, VECF4_ARG tol)
{
    uint32x4_t v = vecf4_near_equal(v0, v1, tol);
    uint32x2_t qlow = vget_low_u32(v);
    uint32x2_t qhigh = vget_high_u32(v);
    uint32x2_t qtemp = vst_u32(qlow, q);
    uint32x2_t qtemp2 = vrev64_u32(qtemp);
    uint32x2_t res = vst_u32(qtemp, q);
    uint32_t ret = 0;
    vst_lane_u32(&ret, res, 0);
    return (ret == 0);
}

// ...more...
```

CPU
-3 ms

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](https://developer.arm.com/architectures/instruction-sets/simd-isas/neon)

MEMCPY

```
void* memcpy (void* destination, const void* source, size_t num);
```

```
__pld(dest);  
__pld(data);  
  
for (int i = 0; i < count; ++i)  
{  
    float32x4_t v = vld1q_f32(data + i*4);  
    vst1q_f32(dest + i*4, v);  
}
```

CPU
-0.3 ms

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.

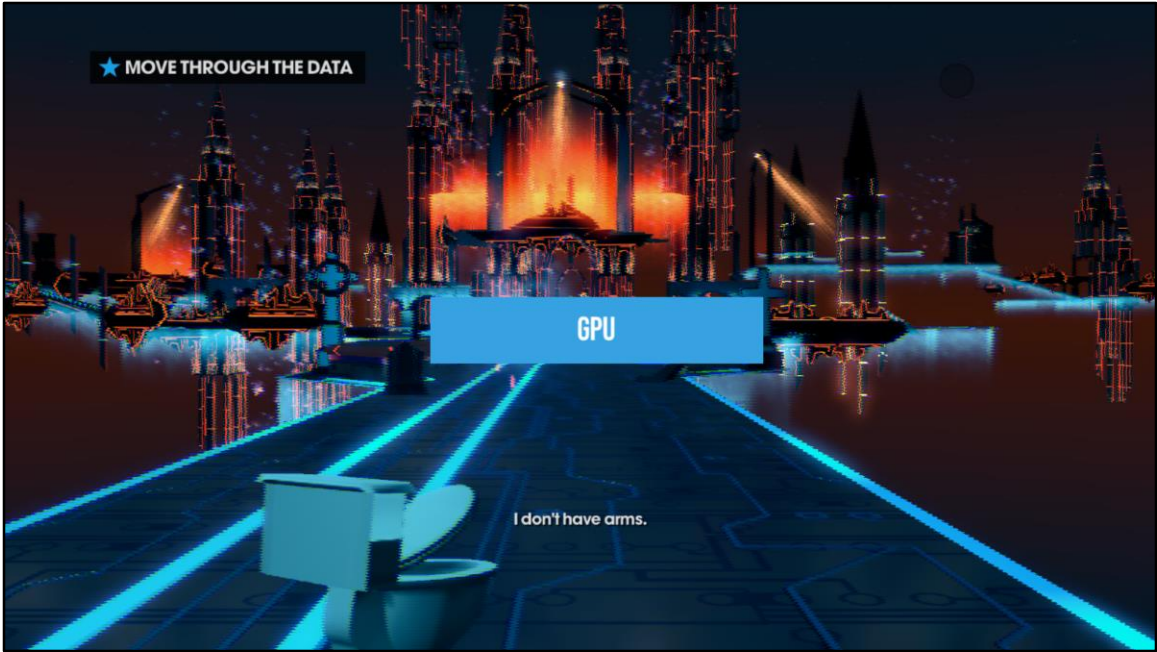
REQUIRES ALIGNED START ADDRESSES AS WELL AS SIZES

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.

AVOID CLEARING

- Especially if not black or white

GPU
-2 ms



**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.**

REDUCE RT RESOLUTION

UNDOCKED

Geometry - 800x450



Materials - 1280x720



End result - 1280x720



DOCKED

Geometry - 800x450



Materials - 1920x1080



End result - 1920x1080



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

CUSTOM MEMORY MANAGEMENT

- Why?
- For what?
- How?

One NVN allocation each:

- Shaders
- Occlusion queries
- Render targets
- Render targets staging
- Textures
- Textures staging
- Meshes

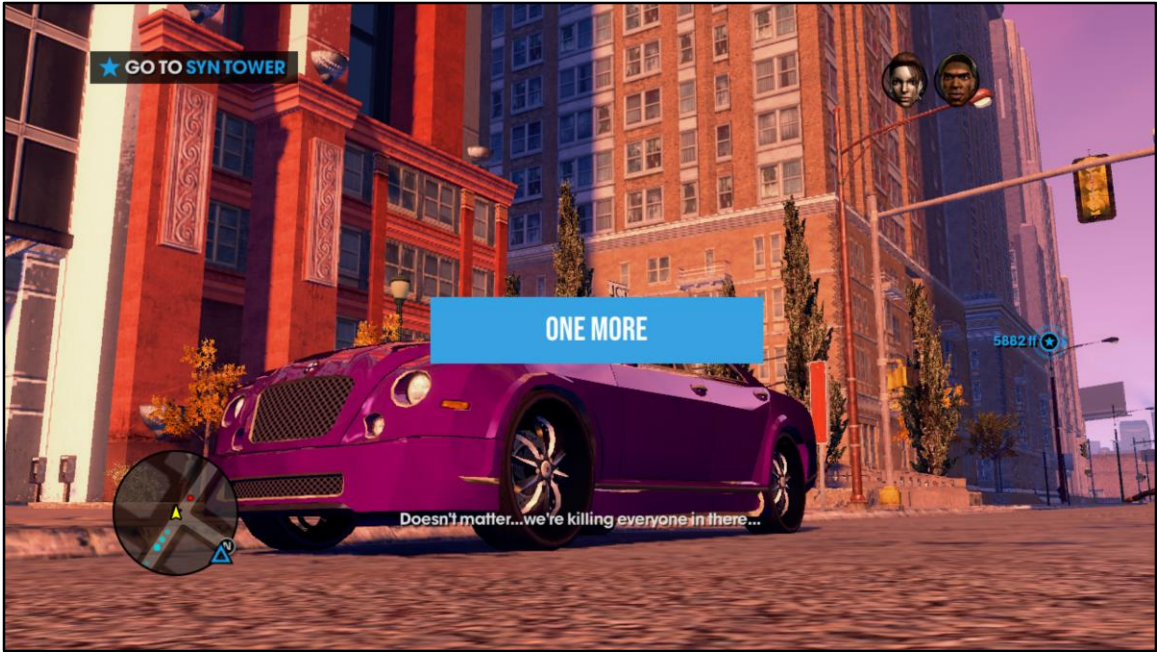


WHY?

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

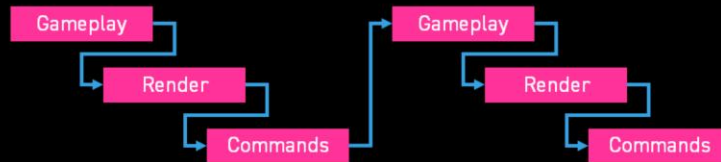
HOW?

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



PARALLELIZATION

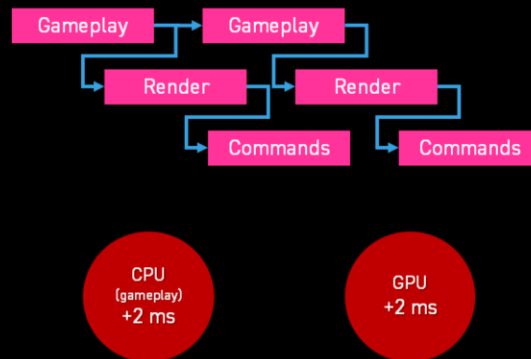
OLD FLOW



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.

PARALLELIZATION

NEW FLOW



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.

PARALLELIZATION IMPLICATIONS

- ▼ Occlusion queries
- ▼ Dynamic meshes
- ▼ Uniforms
- ✔ More time to do stuff



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

OPTIMIZE SIZE

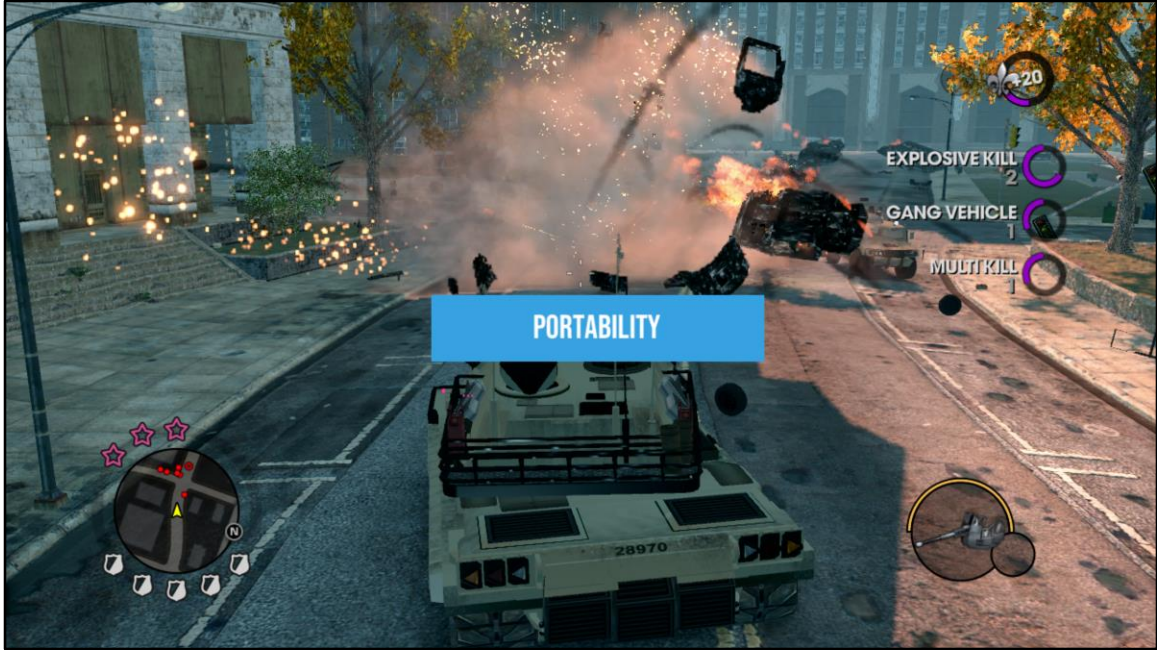


ANOTHER THING WE HAD TO DO WAS TO OPTIMIZE FOR SIZE ON DISK.
THE GAME WAS SUPPOSED TO FIT ON AN 8 GB GAME CARD WITH ALL ITS DLCS.

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.**

THE BIG DECISIONS

- ▼ 3rd party SDKs
- ▼ Mark changes
- ▼ External dependencies
- ▼ Toolchain



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



PLATFORM SPECIFICS

- ▼ Abstract away
- ▼ Searchable markers
- ▼ Fallback implementations
- ▼ #error out



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).

WATCH YOUR LANGUAGE

- ▼ C++ standards
- ▼ Compilers
- ▼ Bits
- ▼ Difficult types



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/>

We're
hiring

© 2019 Deep Silver FISHLABS. Deep Silver FISHLABS is a wholly owned development studio of Koch Media GmbH / Planegg, which is a wholly owned subsidiary of Koch Media GmbH, Austria

