GDC

A New Vehicle Pipeline for *The Last of Us: Part II*

Matthew Trevelyan Johns

Principal Environment Artist. Naughty Dog

Introductions - me

- First experienced game development modding GTA and COD
- University and all things Lego
- Intergalactic Spaceships
- Naughty Dog

Introductions – The Last of Us: Part II

- The sequel to the critically acclaimed ‘The last of Us’.
- Post apocalyptic setting
- Third person, action-adventure/survival horror
- Single player game with an engaging story
- Realistic graphics and animation
- Let’s watch a trailer

Introductions – My Role

- Primarily environment texture and shader artist
- The role is focused on detailed modeling, texturing, shader building and lots of communication

THE LAST OF US PART II

NAUGHTY DOG

Matthew Trevelyan Johns

THE LAST OF US
PART II

NAUGHTY DOG

THE LAST OF US
PART II

NAUGHTY DOG

Matthew Trevelyan Johns

THE LAST OF US PART II

NAUGHTY DOG

Matthew Trevelyan Johns

THE LAST OF US PART II

NAUGHTY DOG

Matthew Trevelyan Johns

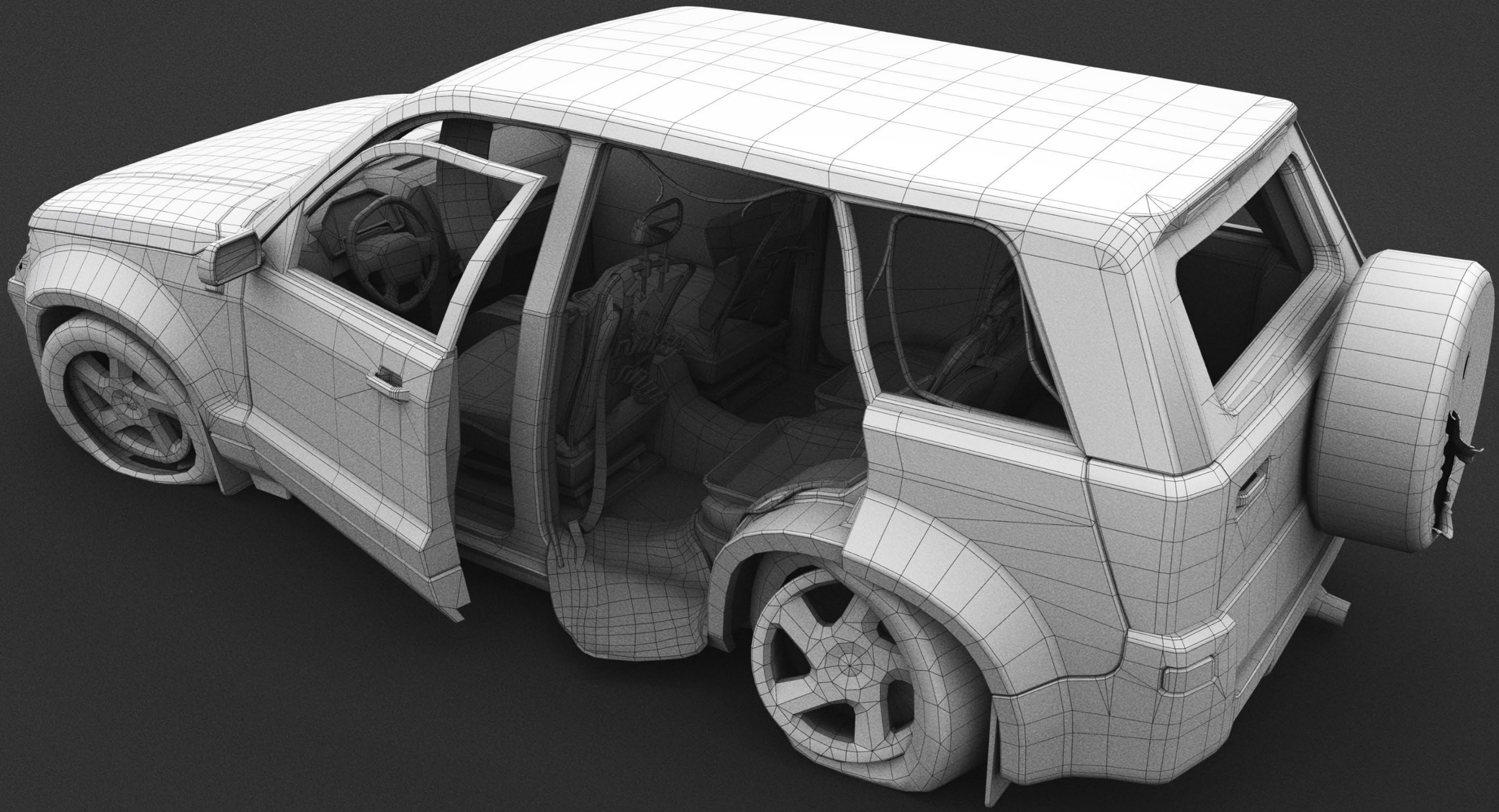
THE LAST OF US PART II

NAUGHTY DOG

Matthew Trevelyan Johns

THE LAST OF US PART II

NAUGHTY DOG



Matthew Trevelyan Johns

THE LAST OF US PART II

NAUGHTY DOG

Introductions – This talk

- It's all about vehicles, but mostly texturing and shading these vehicles
- I speak from my humble artist's point of view
- Concept art, current methods, new methods?
- This is just one of many approaches and not the ONLY approach to the problems we faced

Concept art describes the challenges ahead

THE LAST OF US
PART II

NAUGHTY DOG

THE LAST OF US
PART II

NAUGHTY DOG


THE LAST OF US
PART II
NAUGHTY DOG

GROMANN

THE LAST OF US
PART II

NAUGHTY DOG

THE LAST OF US
PART II

NAUGHTY DOG

THE LAST OF US PART II

NAUGHTY DOG

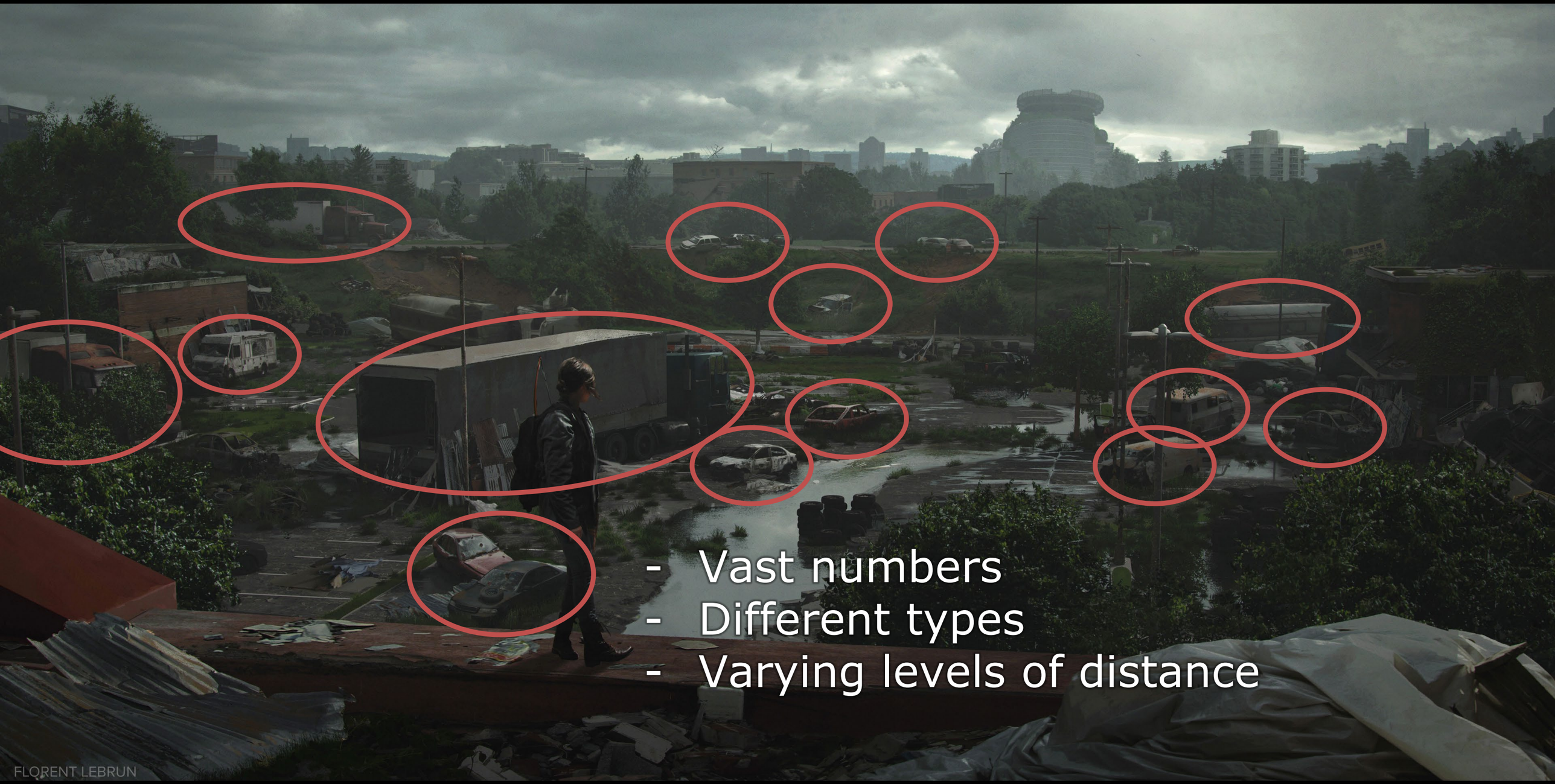

‘Posed’ or articulated doors and hood
Fresh charring and fire damage
GROMANN


Burned vehicle
with moss
Burned vehicles with
water damage


General weathering example
Small leaf piles
Openable trunk/doors
EXIT 2C
NORTH
5
Seattle
EXIT 3A
SOUTH
Rainier Ave.
EXIT ONLY
EXIT 3B
NORTH
Rainier Ave.
SOUTH
Portland
LEFT 3/4 MILE

BALÁZS ÁGOSTON

THE LAST OF US PART II



FLORENT LEBRUN

THE LAST OF US PART II

NAUGHTY DOG

Outline of the major challenges

- Three major requirement categories

- TEXTURING – look, feel, reaction to the world
- MODELING – vehicle types, articulation, interiors
- RENDERING – vast numbers, varying distances, levels of detail?

Concept round up

- A gradually evolving process
- Building upon a foundation other artists had contributed to
- The trailer work was a great way to get practical experience of the challenges we would face
- A long road ahead…

Previous vehicle pipelines at Naughty Dog

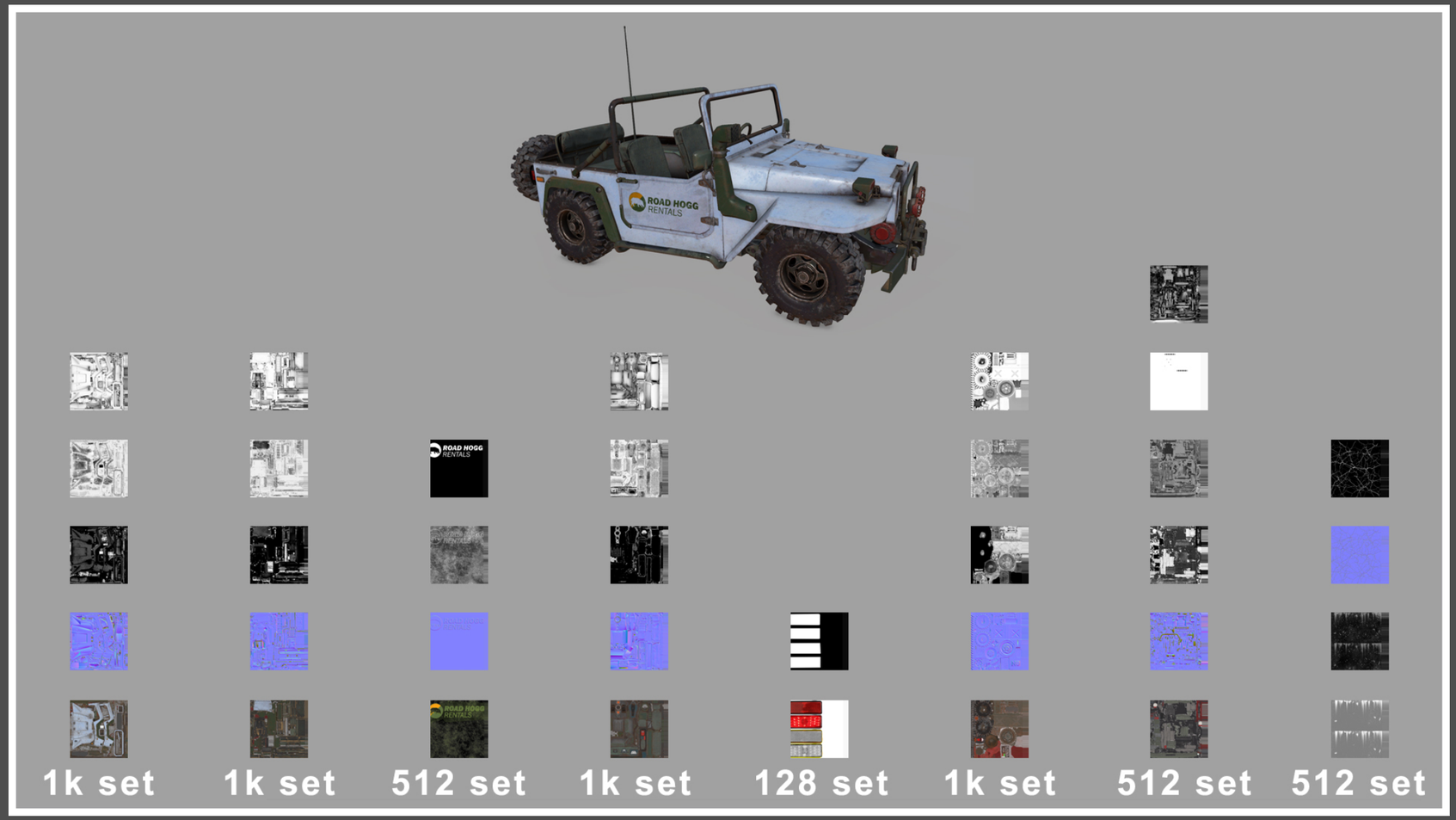
- Examples from Uncharted 4

Example of a hero vehicle in Uncharted 4

- The Madagascar off-road vehicle was created by one of Naughty Dog's awesome artists, Inkyo Lee
- Created using the traditional method: high – low – unwrap – bake – texture – shaders
- The context of it's use in game demanded high levels of detail and interactivity

Texture usage and memory

- Larger, more detailed assets = more textures
- 512 texel density rule + surfaces needing unique UV space
- Each section requires its own texture set
- Baking the off-road vehicle in sections allows us to maintain 512px per 1m
- Let's choose another vehicle from Uncharted 4 that is used in a similar context to those in The Last of Us: Part II



Example of a background vehicle in Uncharted 4

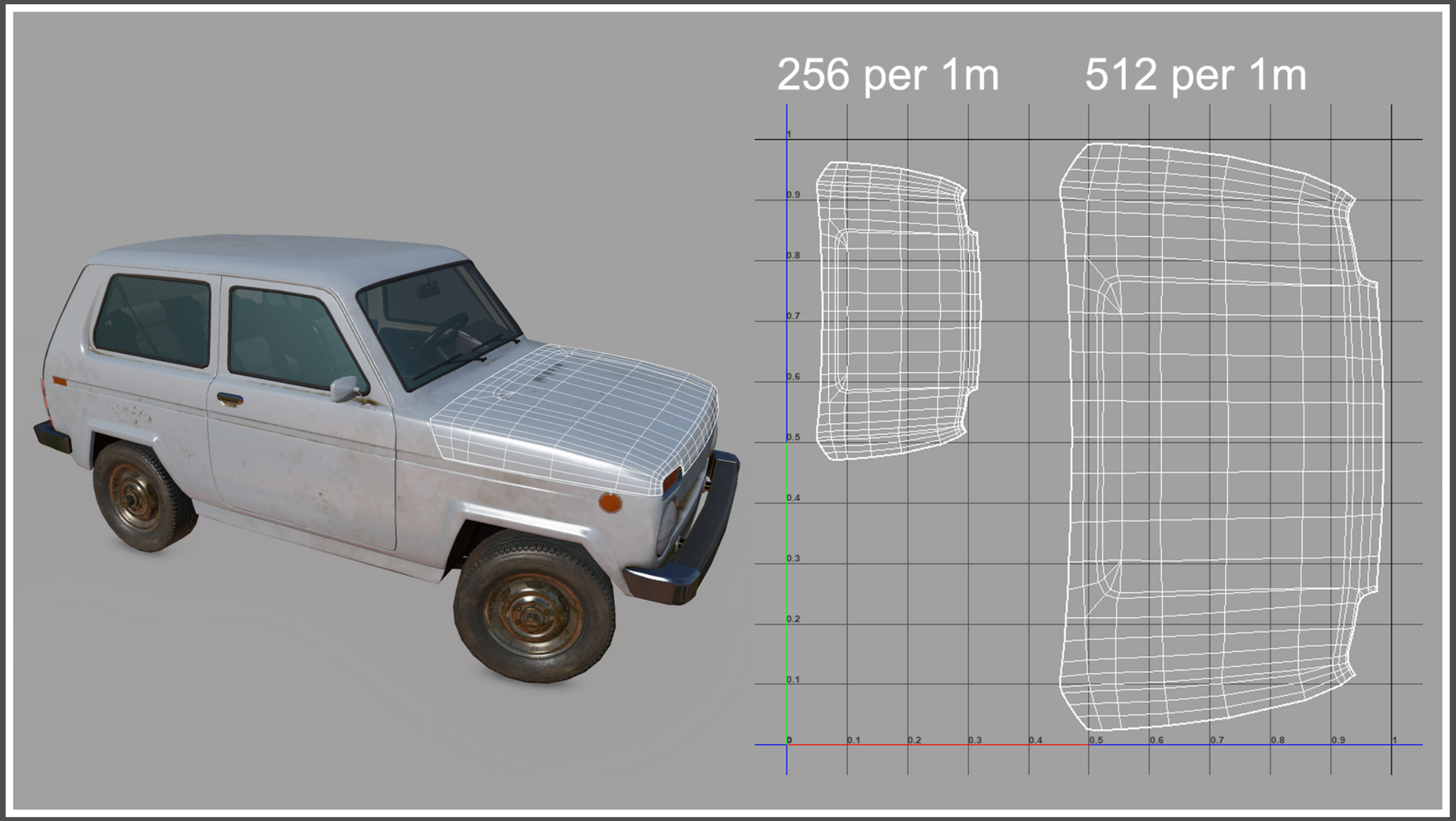
- Background vehicles are a better use-case to examine
- This SUV is representative of the pipeline used

Texel density

- Majority of vehicle mapped at just 256 px per meter, using lots of mirroring.

Texel density cont...

- Mapped to 512 px per meter the exterior alone would require 3 additional texture sets.
- Around 15 more textures (ambient occlusion, normal, color, roughness, metallic)



Texture sets

- 3 x unique texture sets
- 2 x shared texture sets
- 3 x unique masks

Larger vehicles

- As predicted, larger vehicles require more texture sets
- Dirt and grime was also included in the vans base textures…

Resolution and details

- However, results are quite low resolution due to the 256 pixels per meter mapping resolution

Texture memory implications

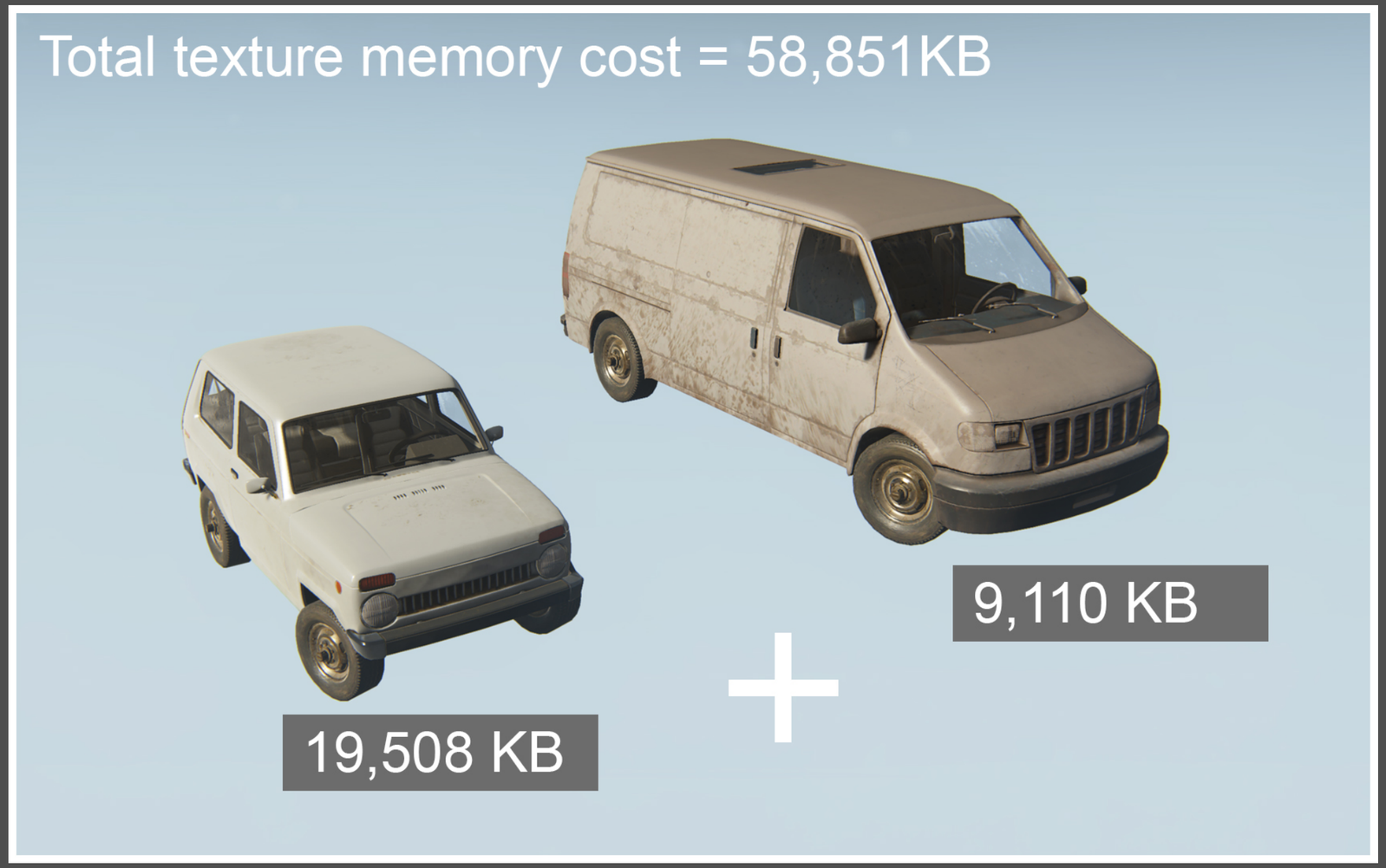
- Why do a few more textures even matter?
- We tend to work around a fixed limit and try to squeeze as much out of it as possible
- We can consider 30MB our ‘baseline’

Texture memory implications

- The SUV model costs us 19,508 KB

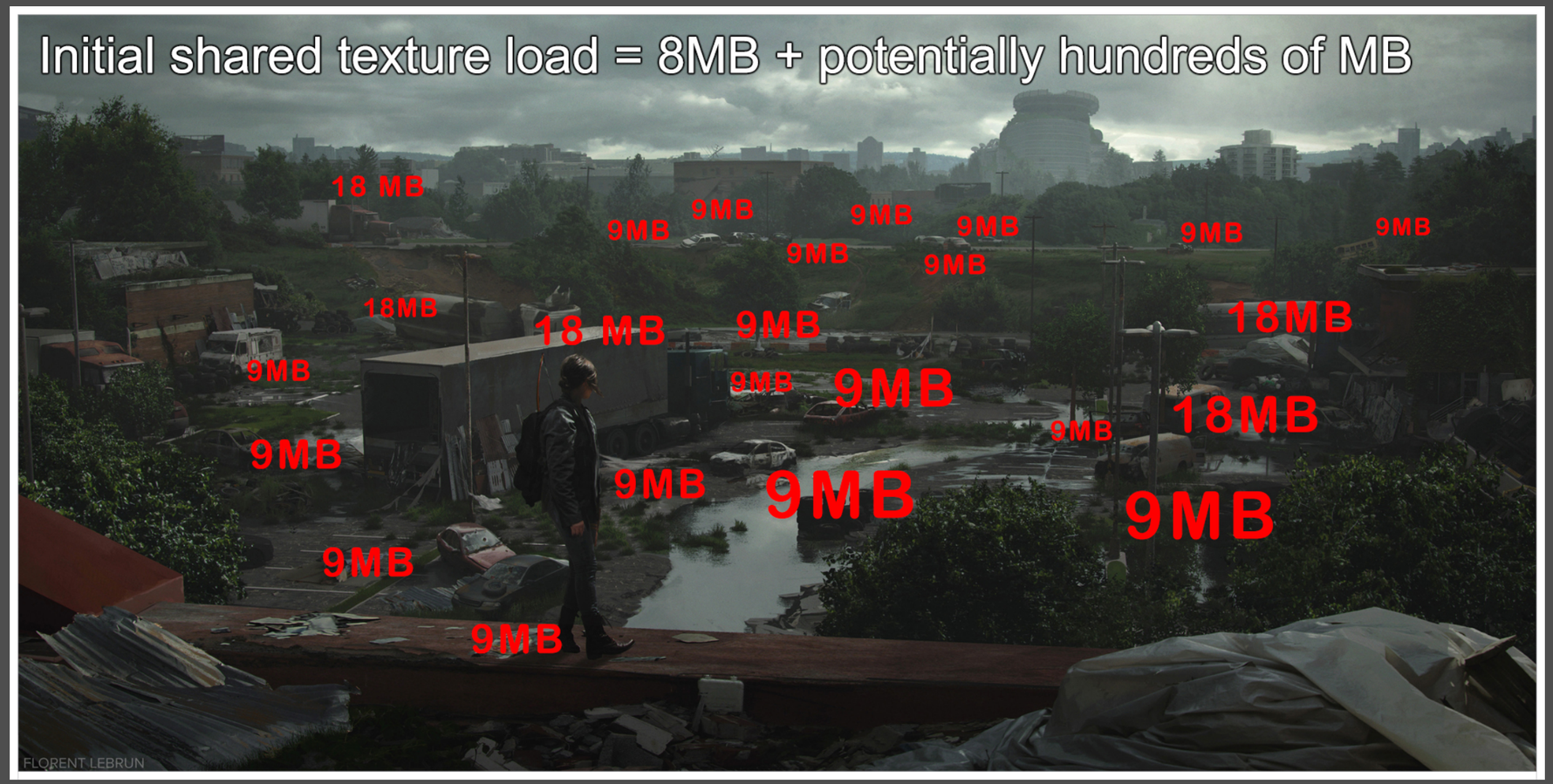
Texture memory implications

- The van costs us an additional 9,110 KB of texture memory
- Heavy texture memory cost for every new vehicle added



Texture memory implications

- Texture memory getting eaten up by all these vehicles
- Vehicles will use texture memory, but it can't be so much that its detrimental to environment artists
- Building upon foundations established in previous games and upon ideas from other Naughty Dog Artists
- Let's (finally) talk about the new vehicle pipeline!

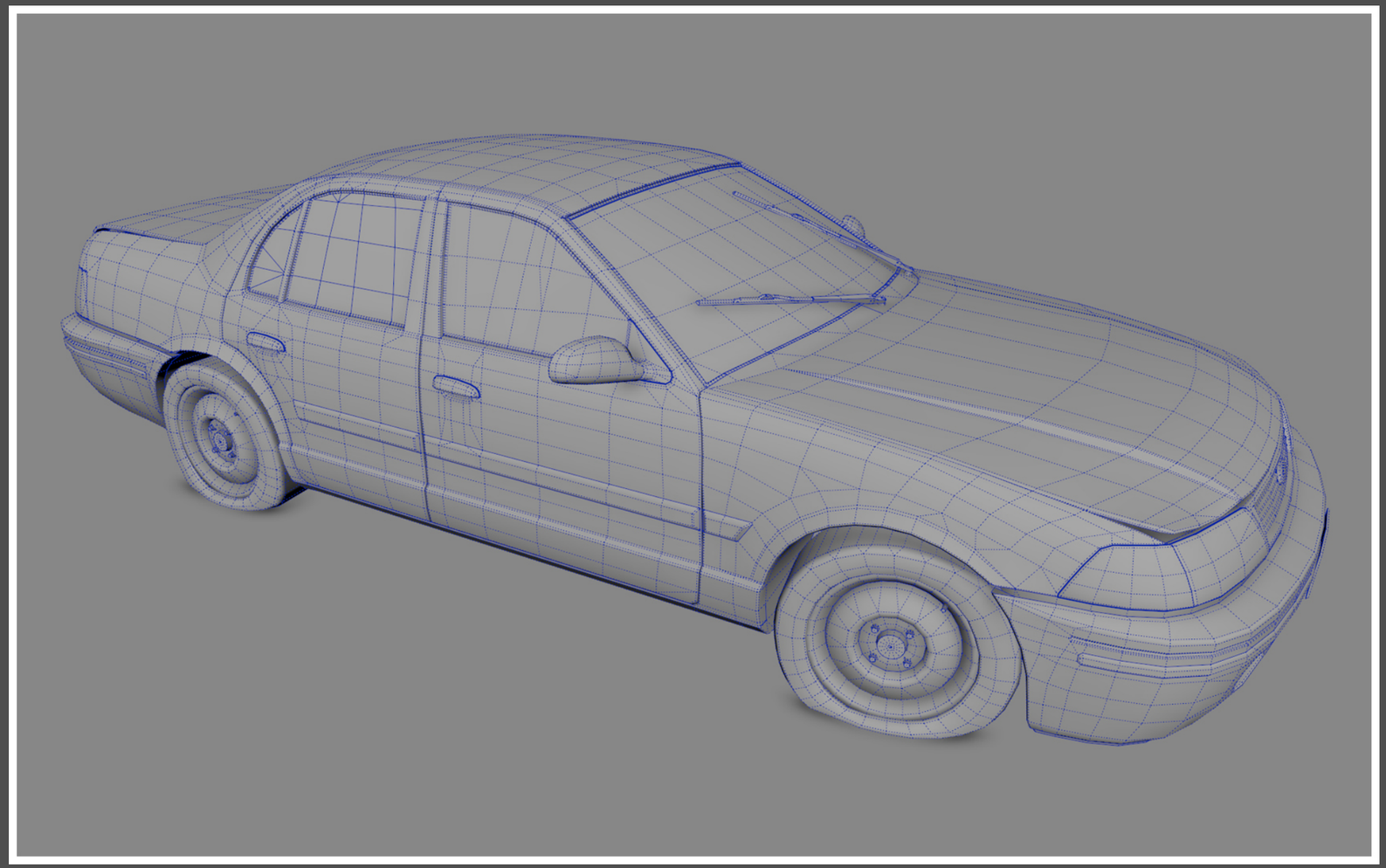


The new vehicle pipeline

- Full art process, from start to finish
- Advantages of this new pipeline?
- How does this save texture memory?

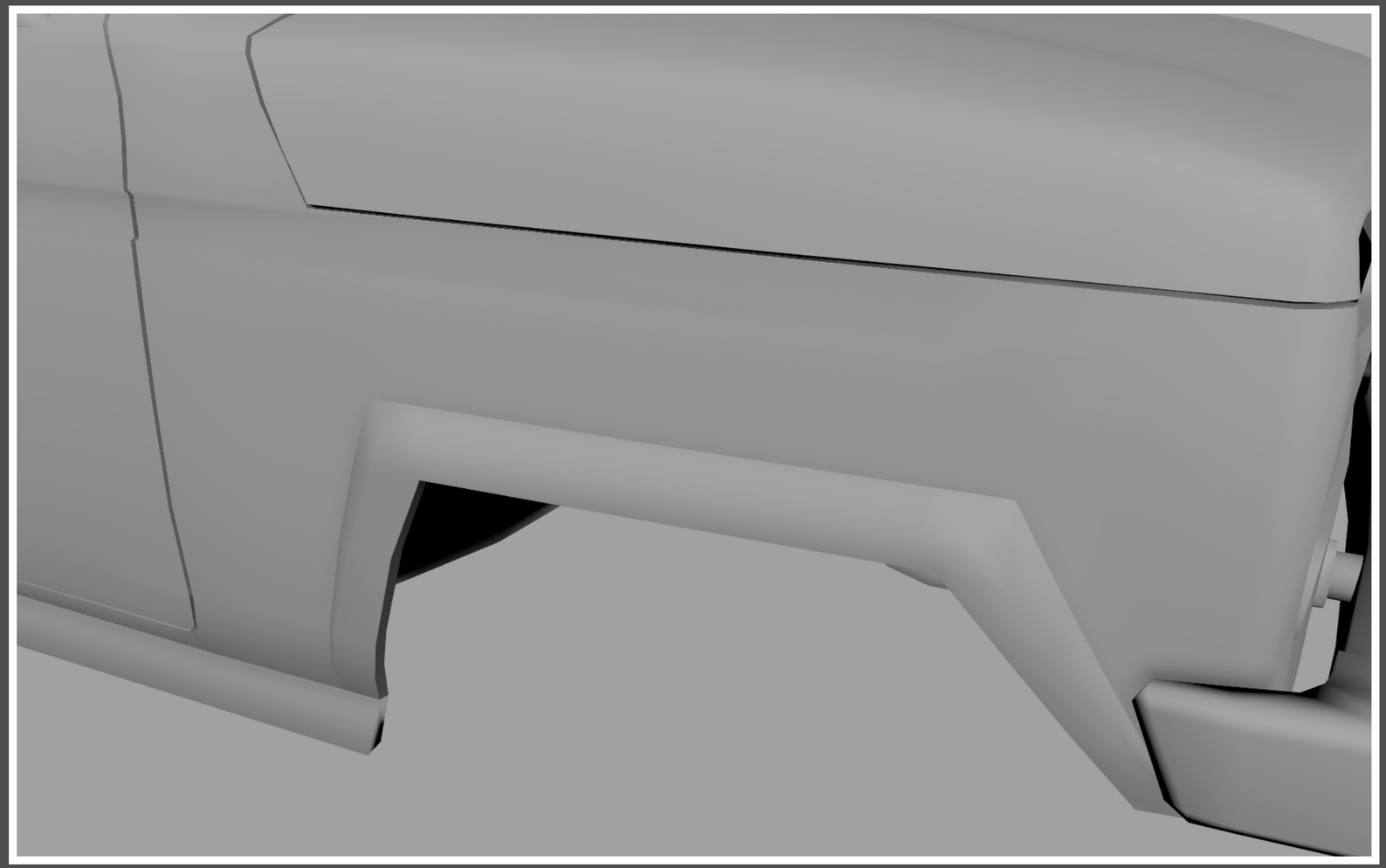
Modeling basics

- Drop baked normals for the body completely
- This removes an entire stage of outsource production that was time consuming
- It also immediately saved a lot of unique textures per vehicle
- Baked normals for subtle body shape details aren't necessary
- Sometimes, only a slight increase in polygons



Baked shape details vs modeled

- Low poly, no bevels, hard edges

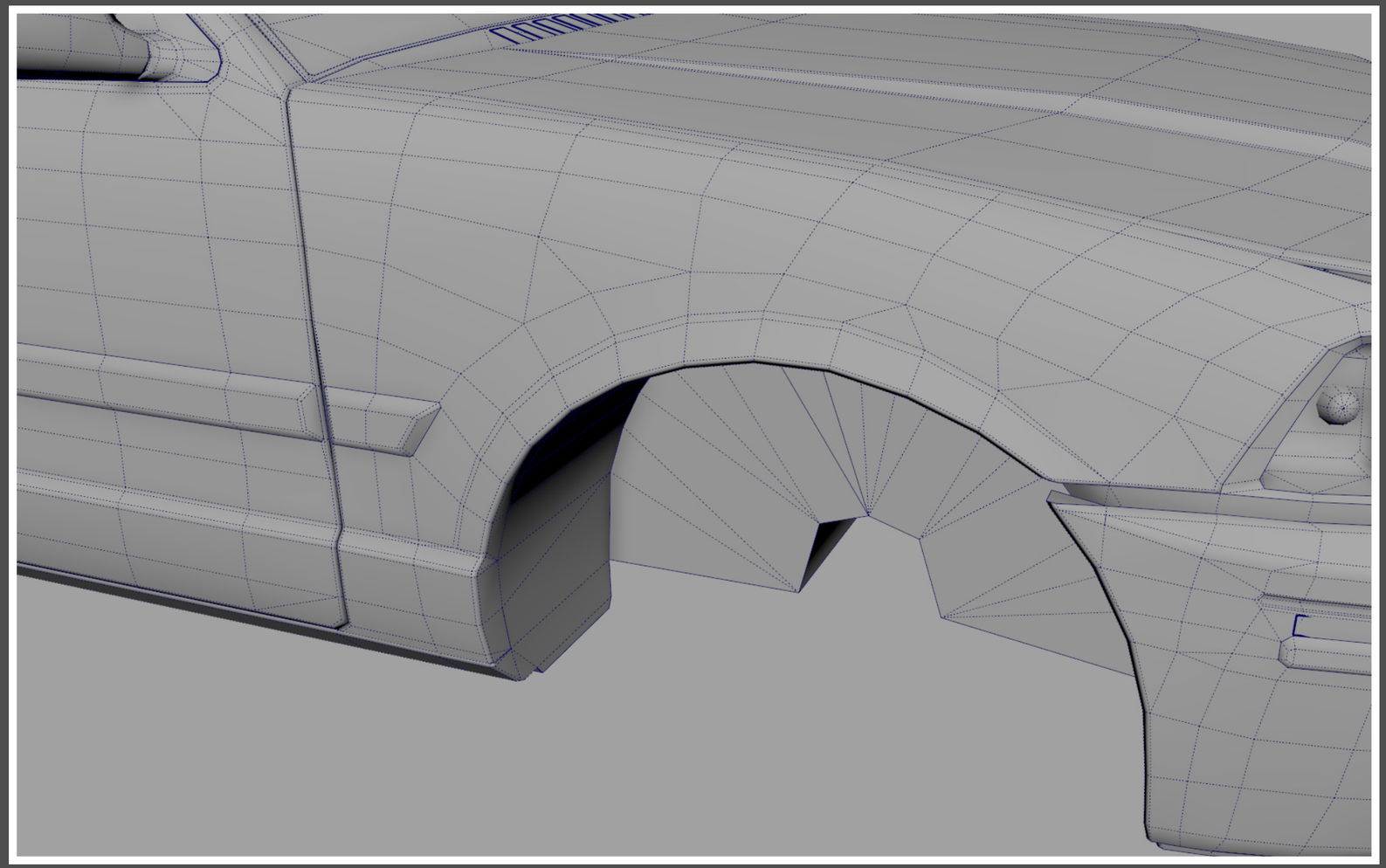


Baked shape details vs modeled

- Bake adds impression of soft bevels and some additional, low res details

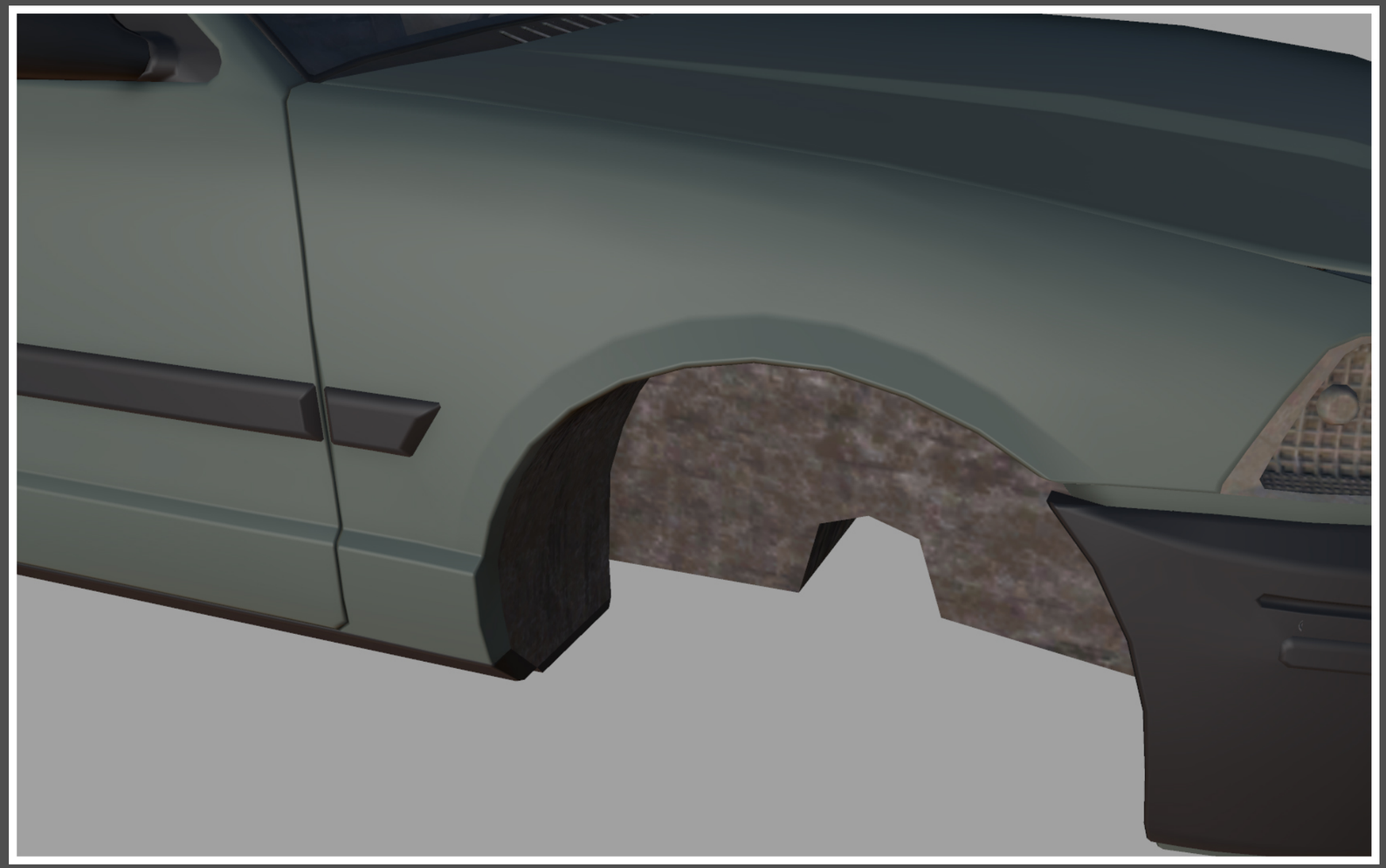
Baked shape details vs modeled

- Higher polycount, bevels, no bakes



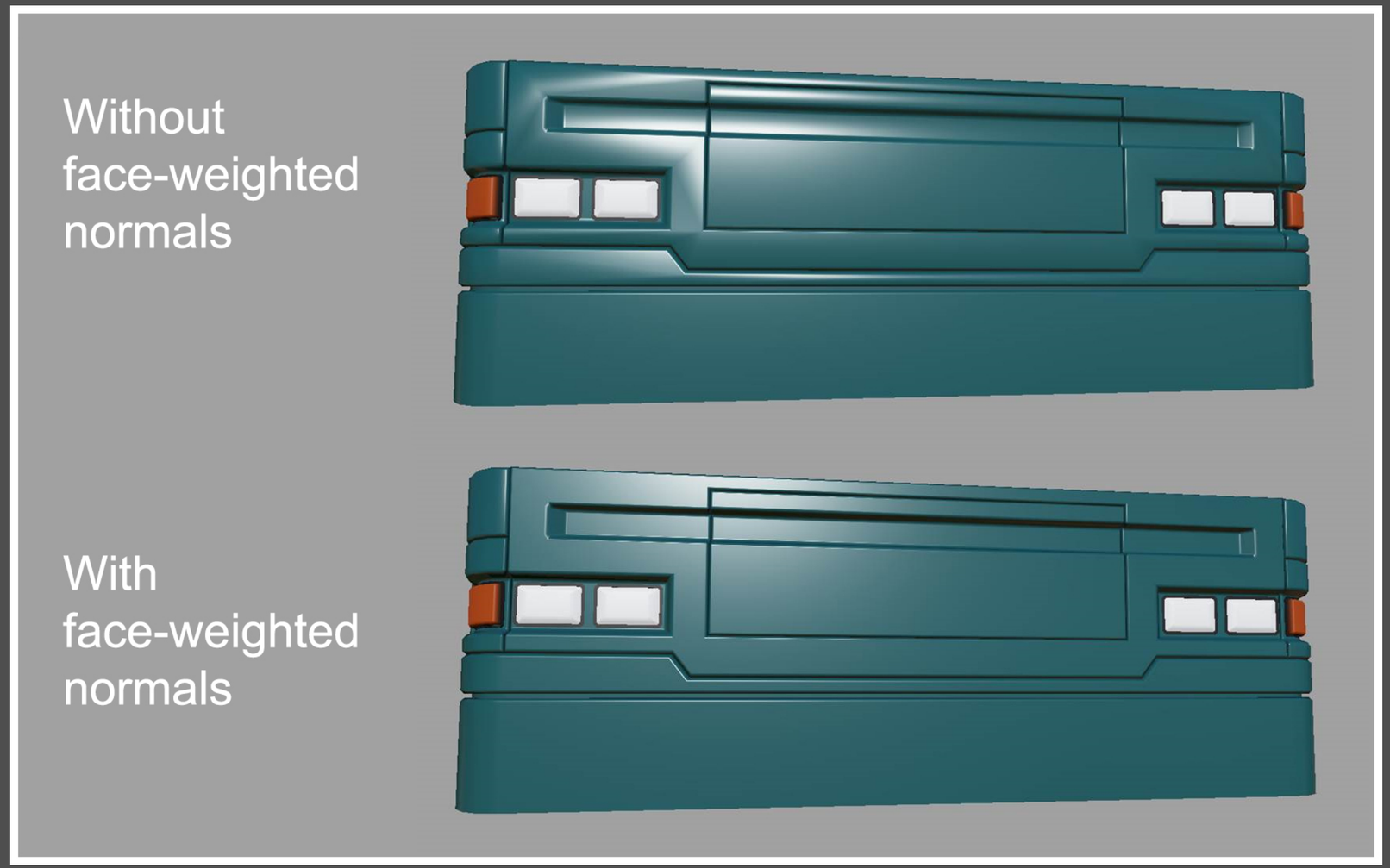
Baked shape details vs modeled

- Resulting model still looks nice and smooth and the bevels light well



Face weighted normals

- Soft, pillowy normals can occur and cause a bad reaction to lighting
- Using face-weighted normals fixes this and our script made it quick and easy

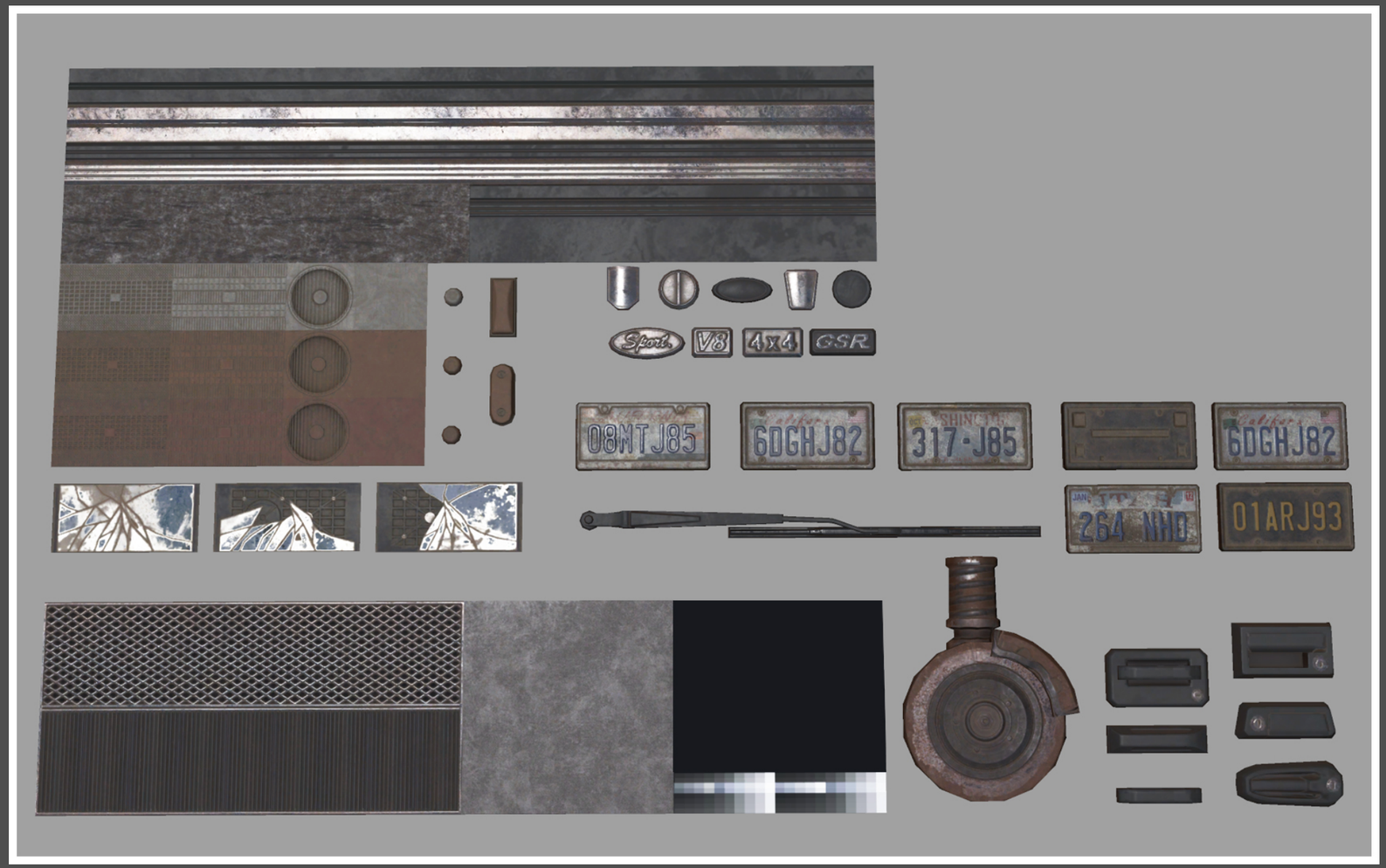


Details

- What about those smaller detailed elements? We can't model everything with higher polycounts…

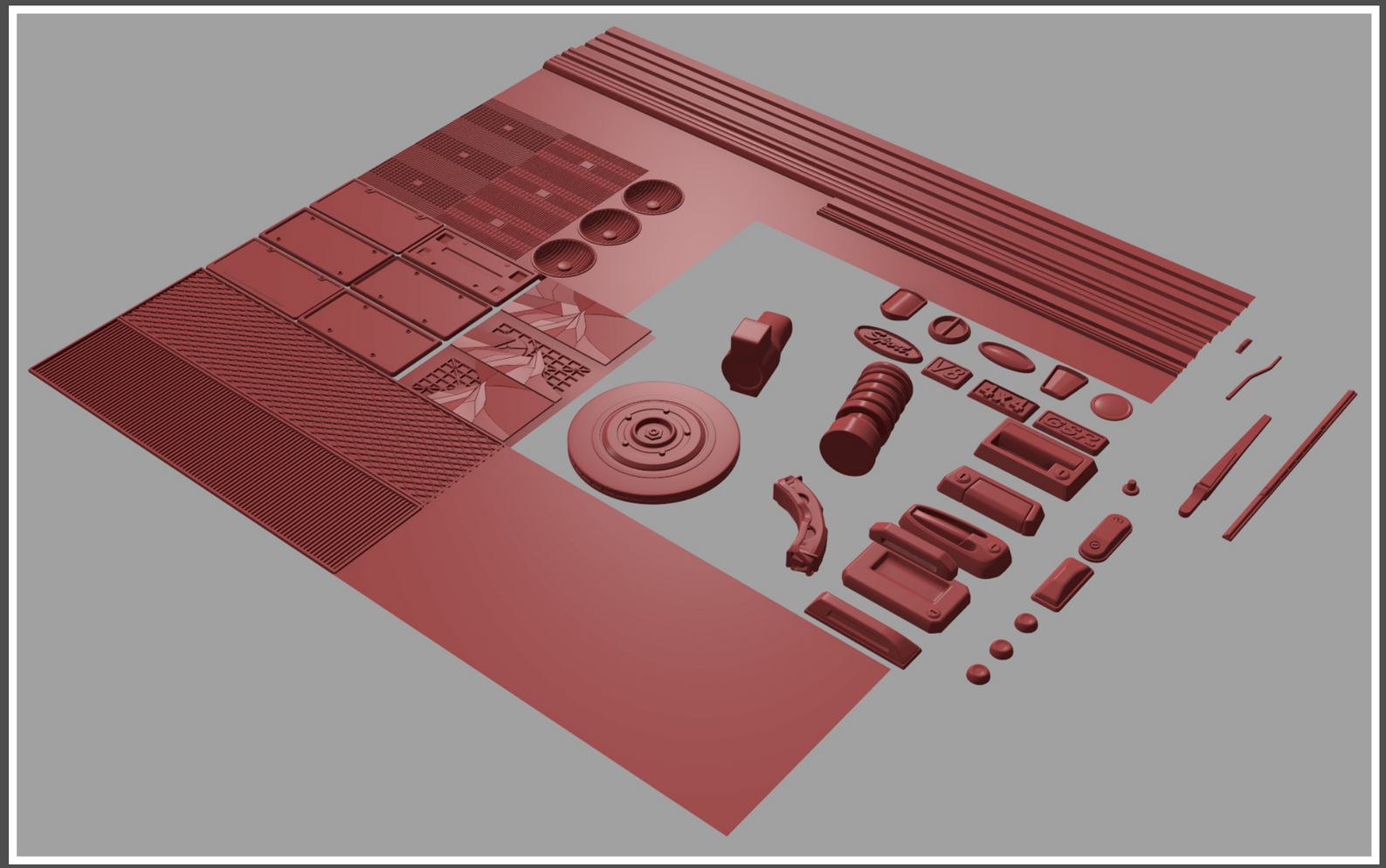
The bake kit

- A collection of modular, generic details baked from a high poly model and textured
- These details are not unique to one car, but suitable for many cars



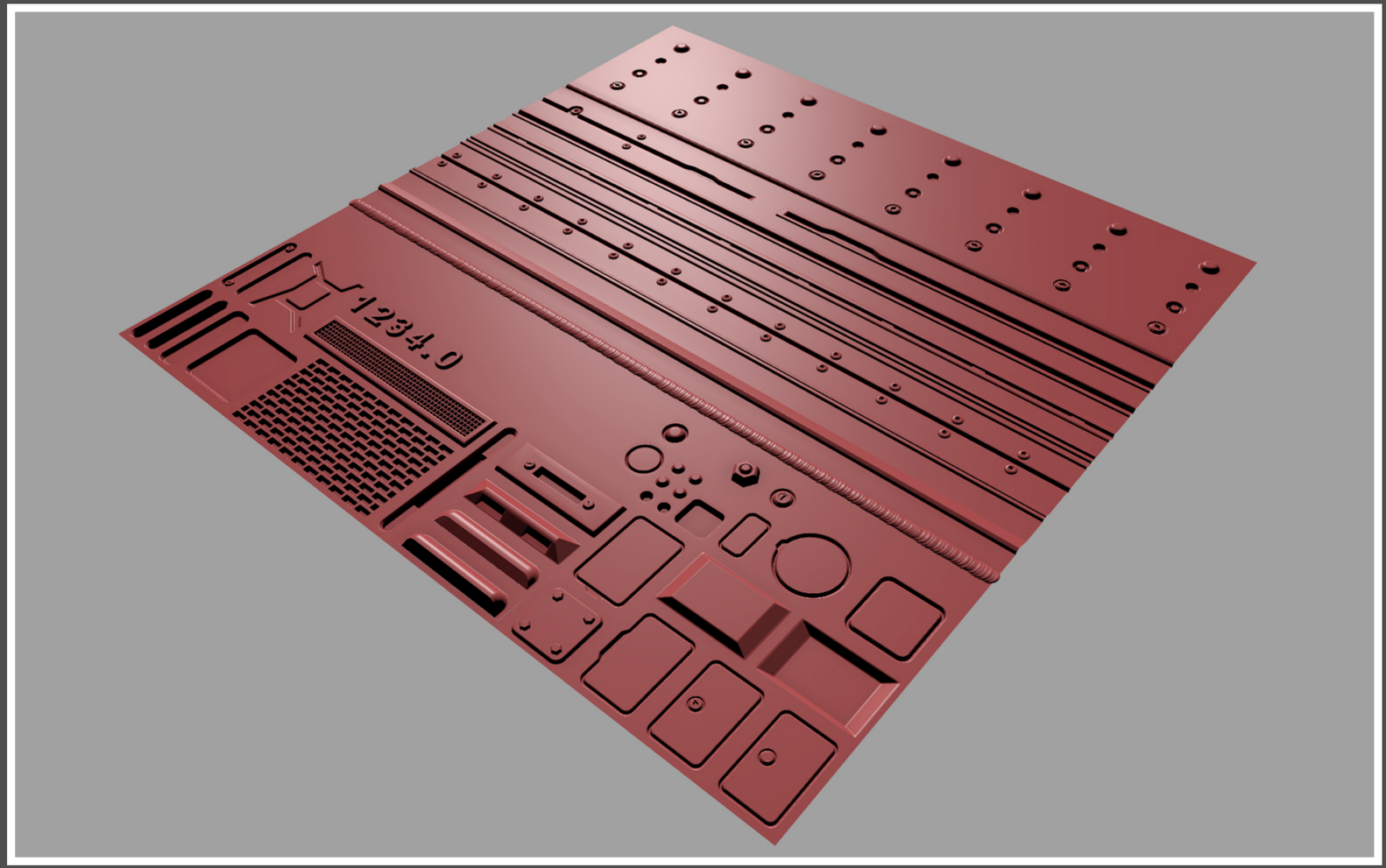
The bake kit

- The baked kit contains elements suitable for multiple small vehicles
- One vehicle might only use a few parts, but across the library of small vehicles, all parts were useful
- Ambient Occlusion is not used in the shader, to save texture memory



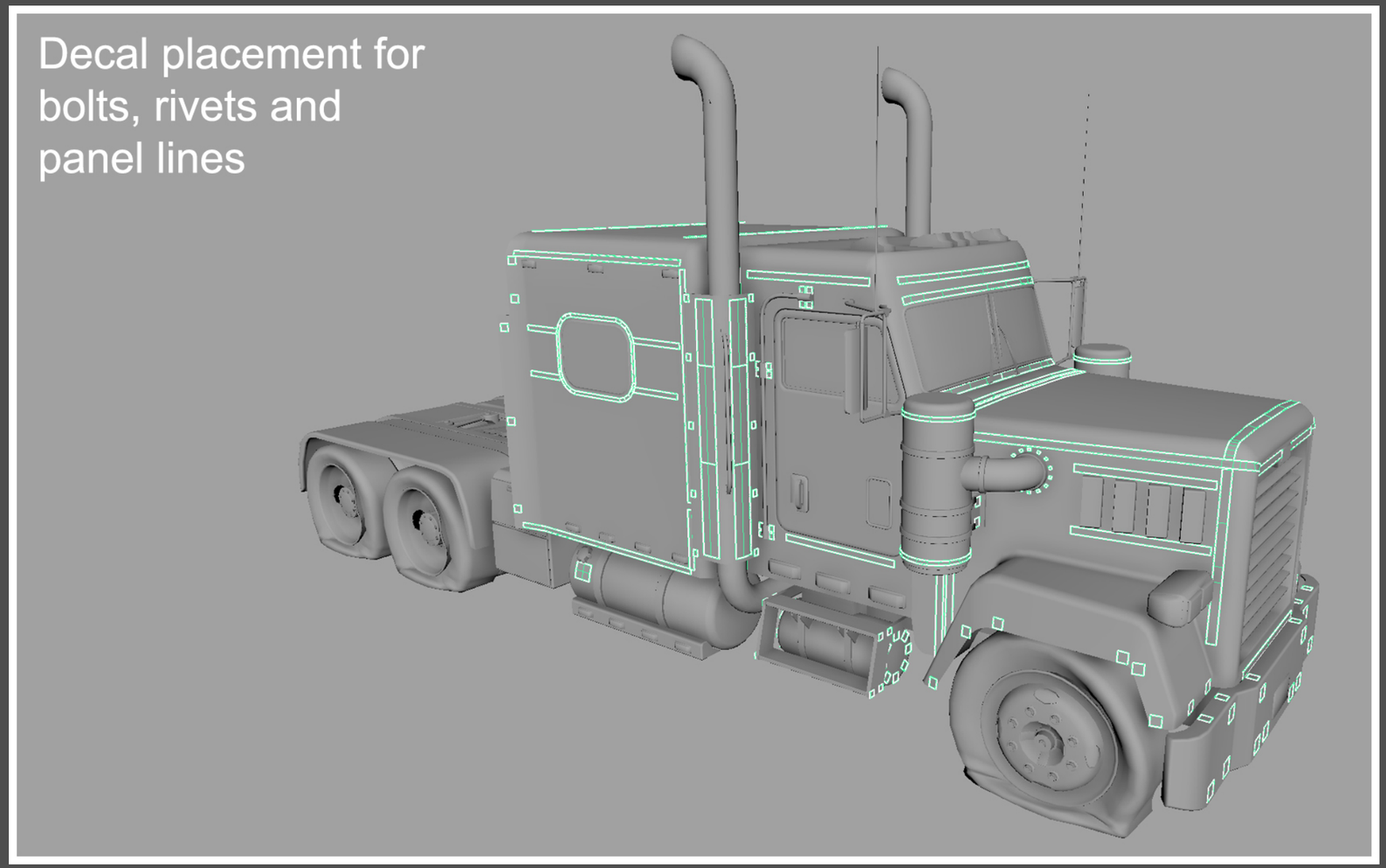
Decals

- Parallax occlusion mapped decals for bolts, vents and other small details
- Height and normal maps do most of the work and the shader inherits other channels from the surfaces below it in order to feel natural



Decals

- Strips were floating above the vehicle surface and were hand placed and uv mapped
- Vertex compression at build time was reduced to prevent z-fighting artefacts



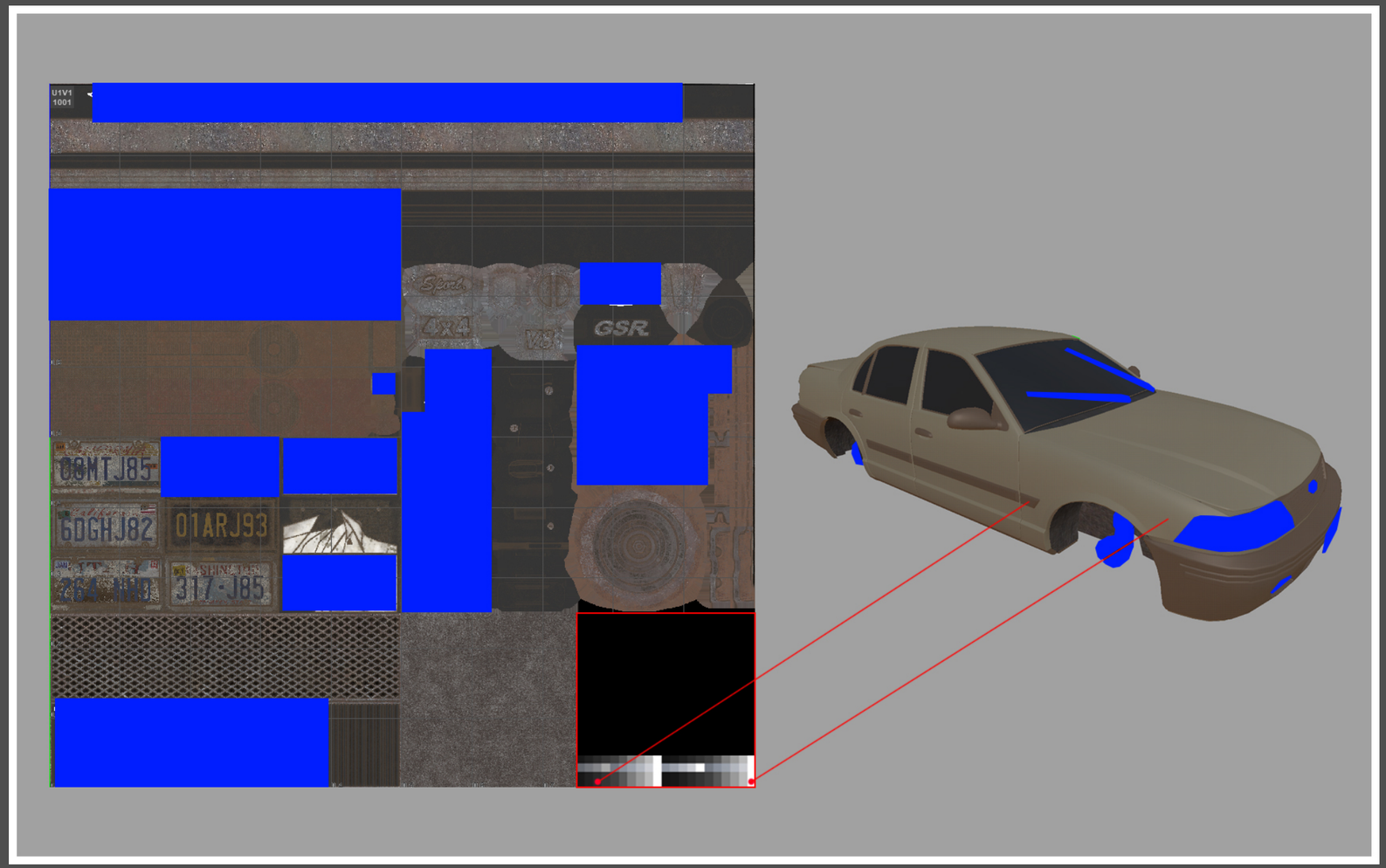
Decals

Decals

- These decals became were really useful for environments as well as vehicles

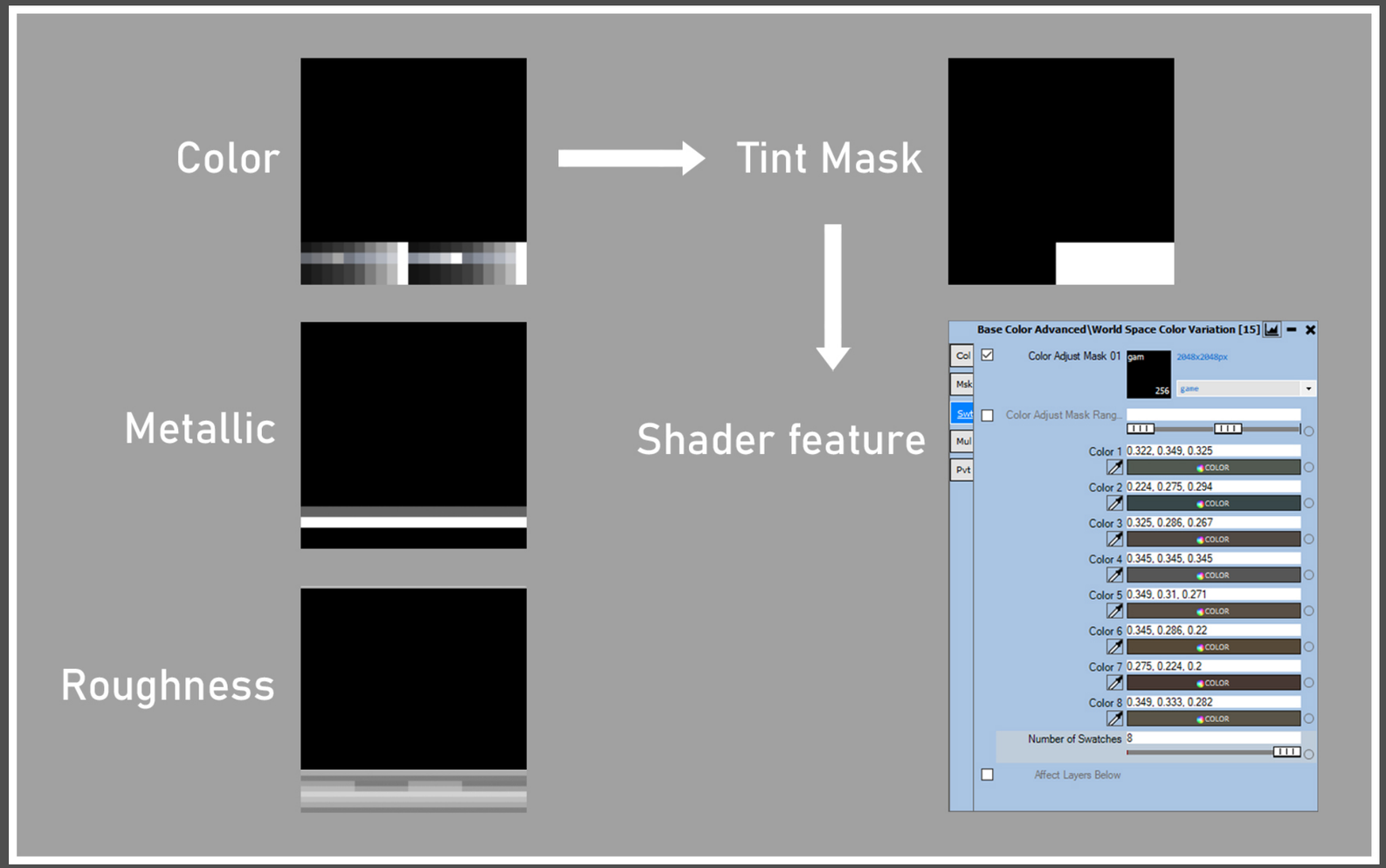
Body colour

- A small swatch texture drives the body colour, roughness and metallic values
- The body UVs are shrunk down and placed on the swatch
- Multiple material types are represented by the swatch



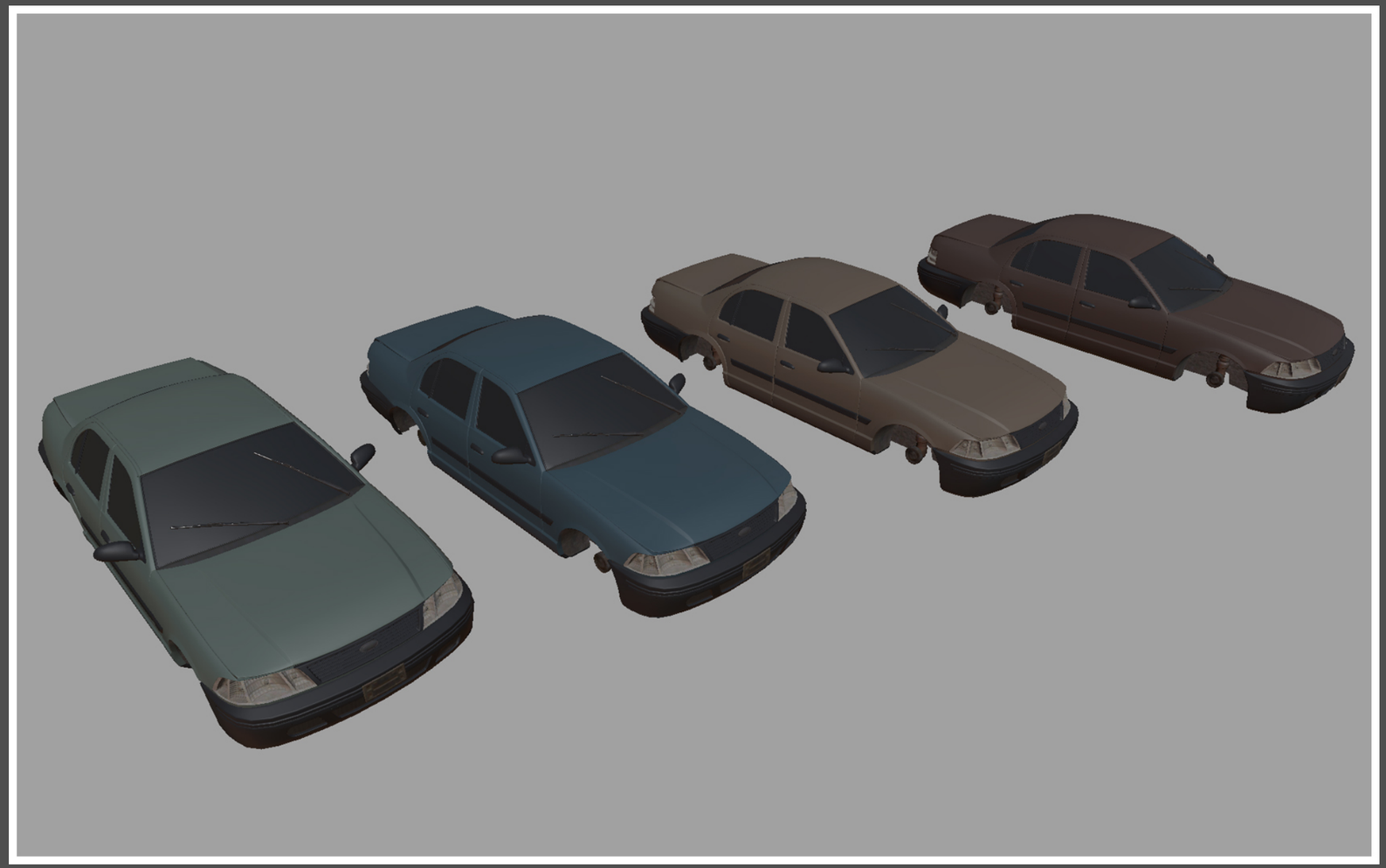
Colour swatch

- We don’t want duplicate cars of the same colour side by side
- A shader feature tints the swatch colours based on the vehicles World space position coordinates



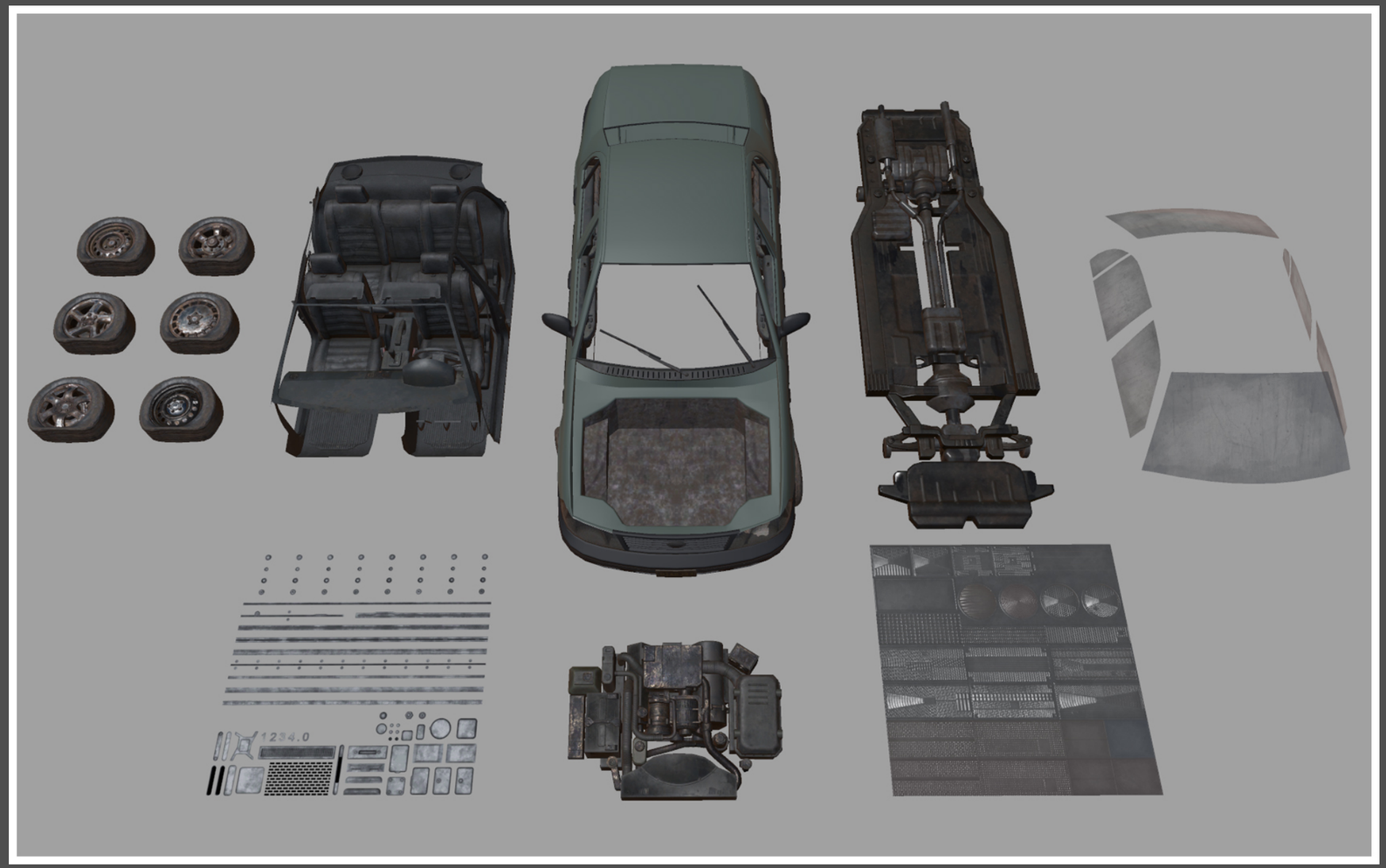
Colour variation

- Each vehicle can have up to 8 different colours
- So far, so good, optimal and using a method that will work for others
- What about the other details we need?



Multiple kits

- All elements are built from shared kits and shared shaders
- Sharing is caring
- All baked from high poly models to unique textures
- Building upon and refining ideas from past projects and other artists
- How do we make up for missing surface detail?

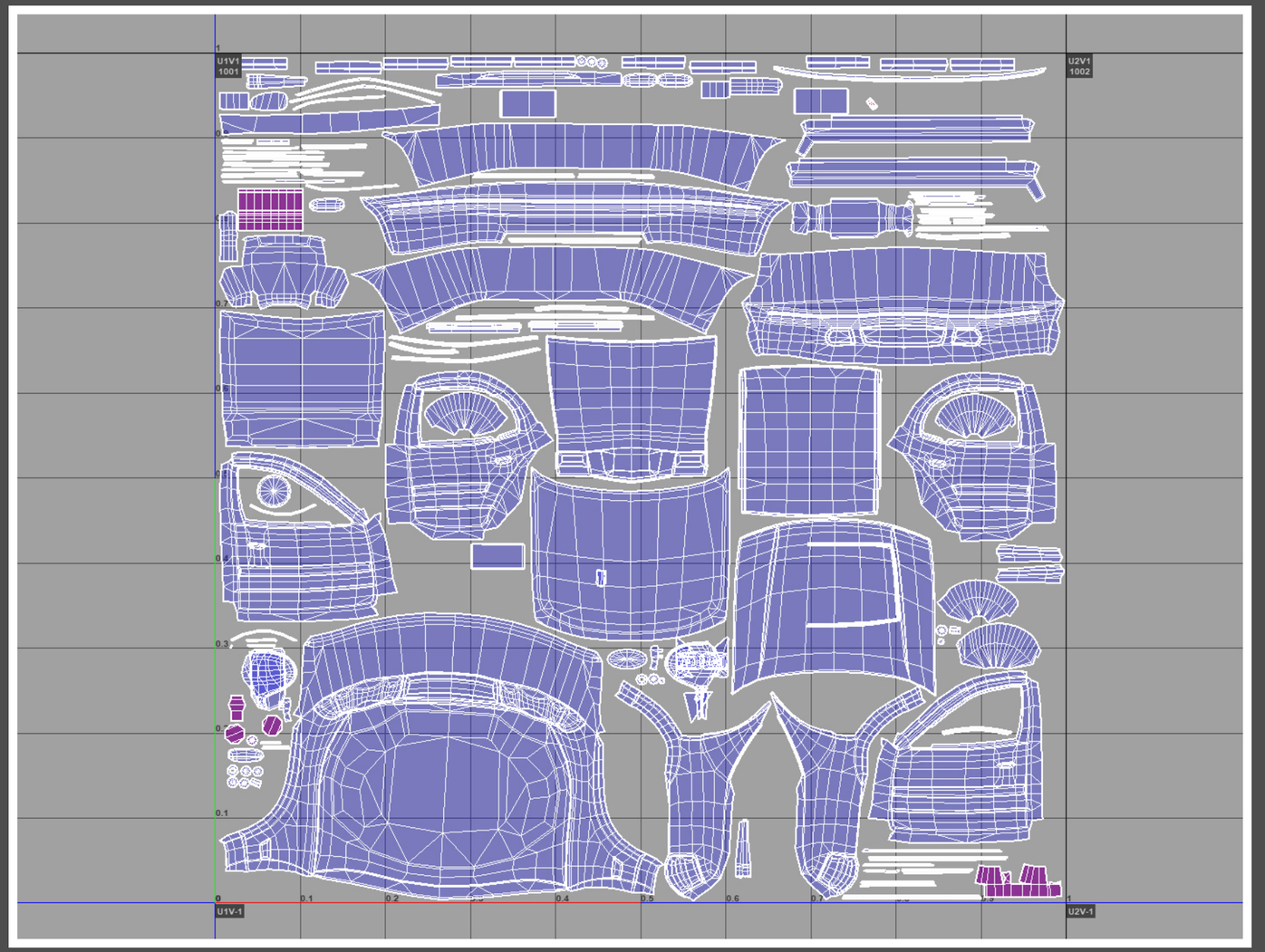


Refining the look

- Still aiming to hit the detailed looks presented by the concept art
- Advanced looks like wetness, moss, snow, burning etc
- Uniquely baking and texturing all of these details isn't possible
- Masks + multi layered shaders + system of shader instancing and texture sharing

Masks

- The vehicle is given a secondary UV set
- UV shells from the first UV set are copied here, uniformly scaled and packed efficiently
- Substance Painter used to paint the masks



Smart material mask

- Uniquely hand painting any textures is time consuming and can be inconsistent
- Creating a template for the masks solves this
- And this consistency will be reflected in the shaders
- Substance Painter smart material
- and alpha library created to be shared with Outsource vendors

Smart material mask

- Semi truck wear-mask exhibits consistent results

Smart material mask

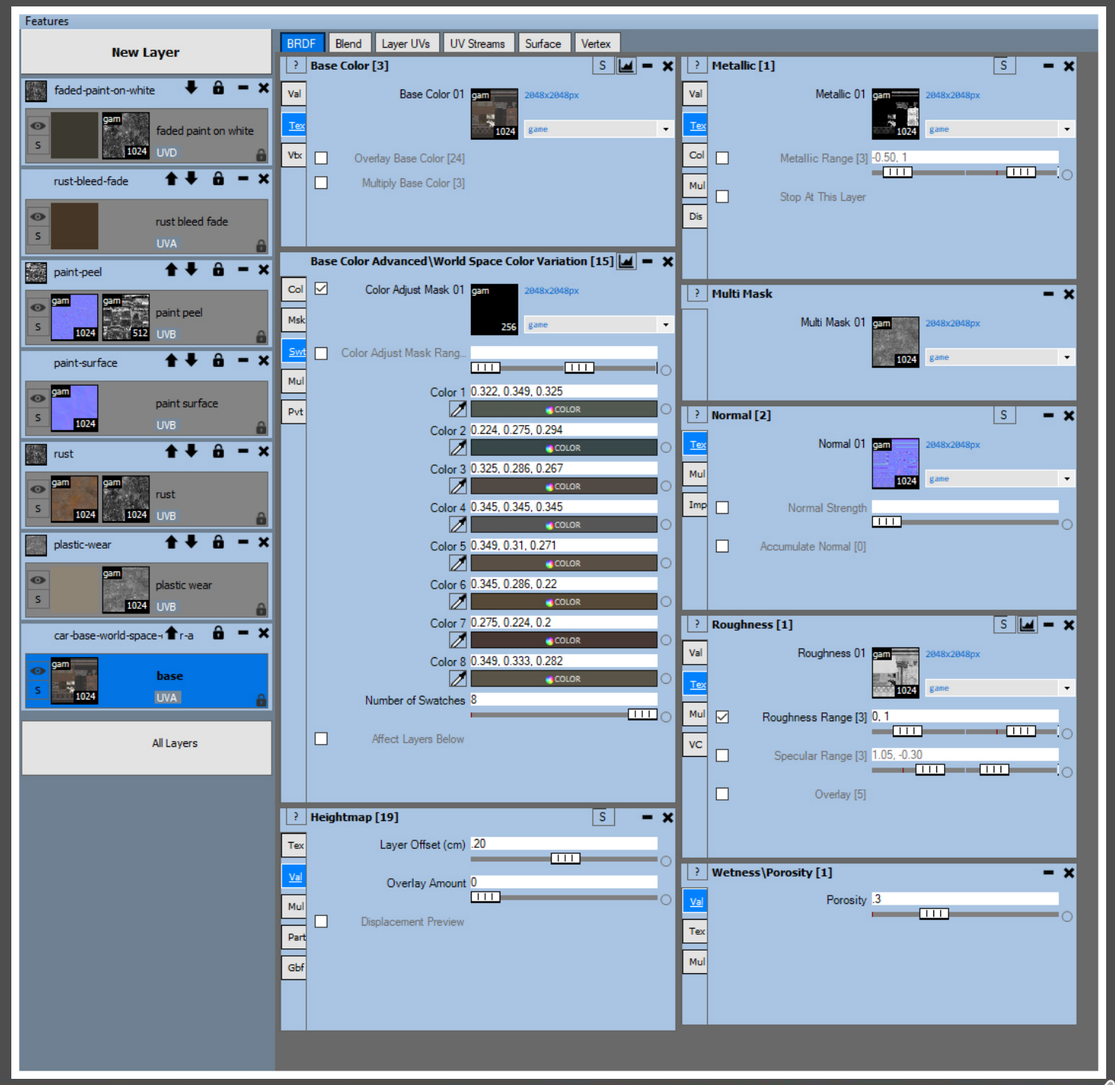
- Drips and small details added with alphas very quickly

Shaders

- A shader layer for each component of ‘the look’ driven by the wear mask and other blend features

Surfer

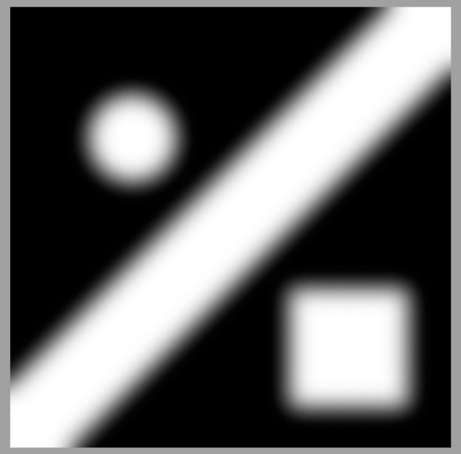
- Our in-house shader editor, showing the BRDF tab and multiple, stacked layers

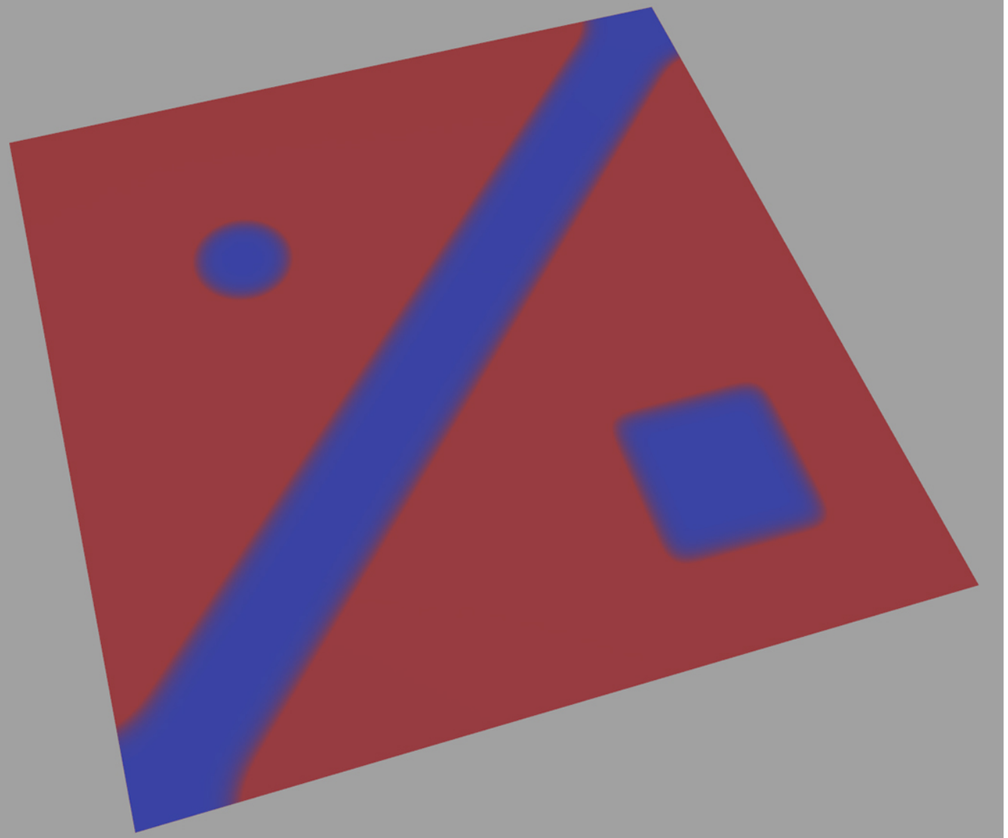


Texture blend

- ‘Texture Blend’ = a black and white mask that blends one layer over another

Texture blend mask

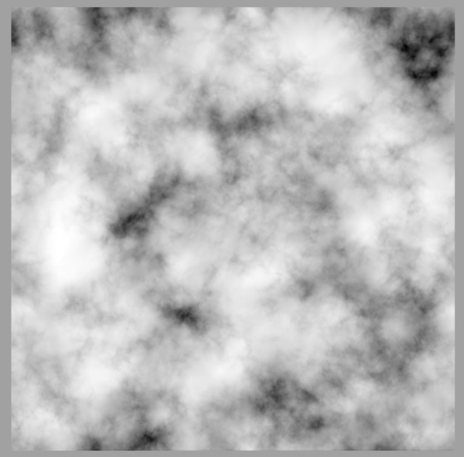


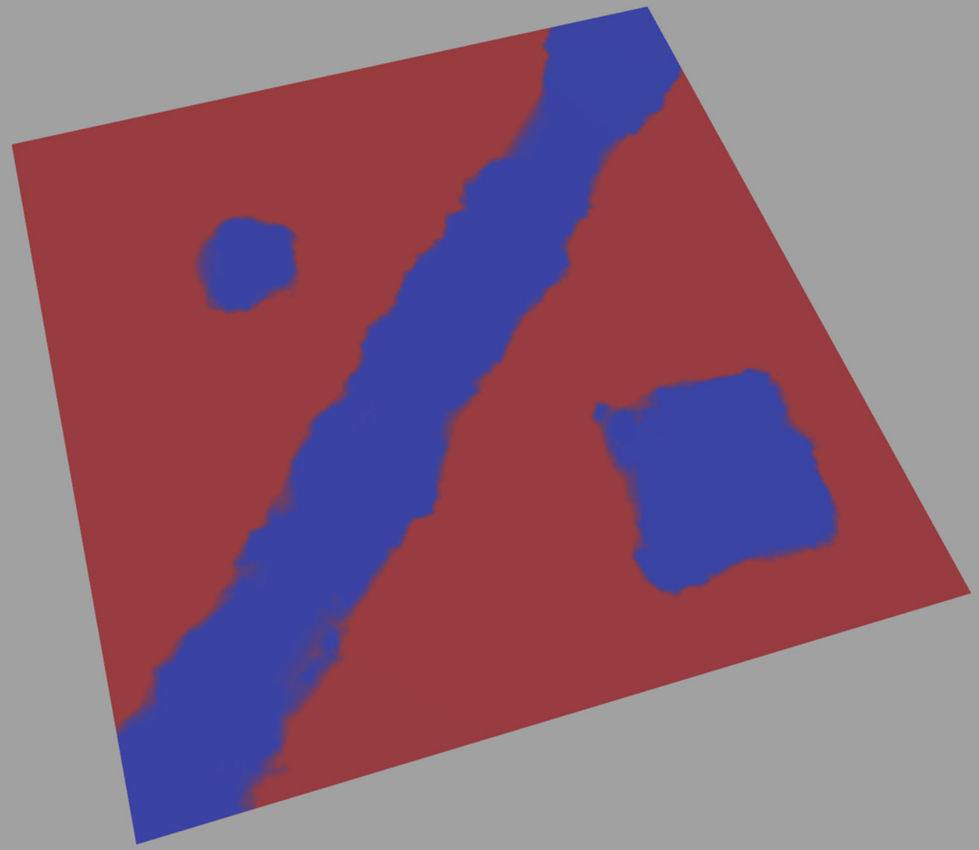


Threshold blend

- ‘Threshold Blend’ = a black and white mask that adds detail at the edge of the texture blend

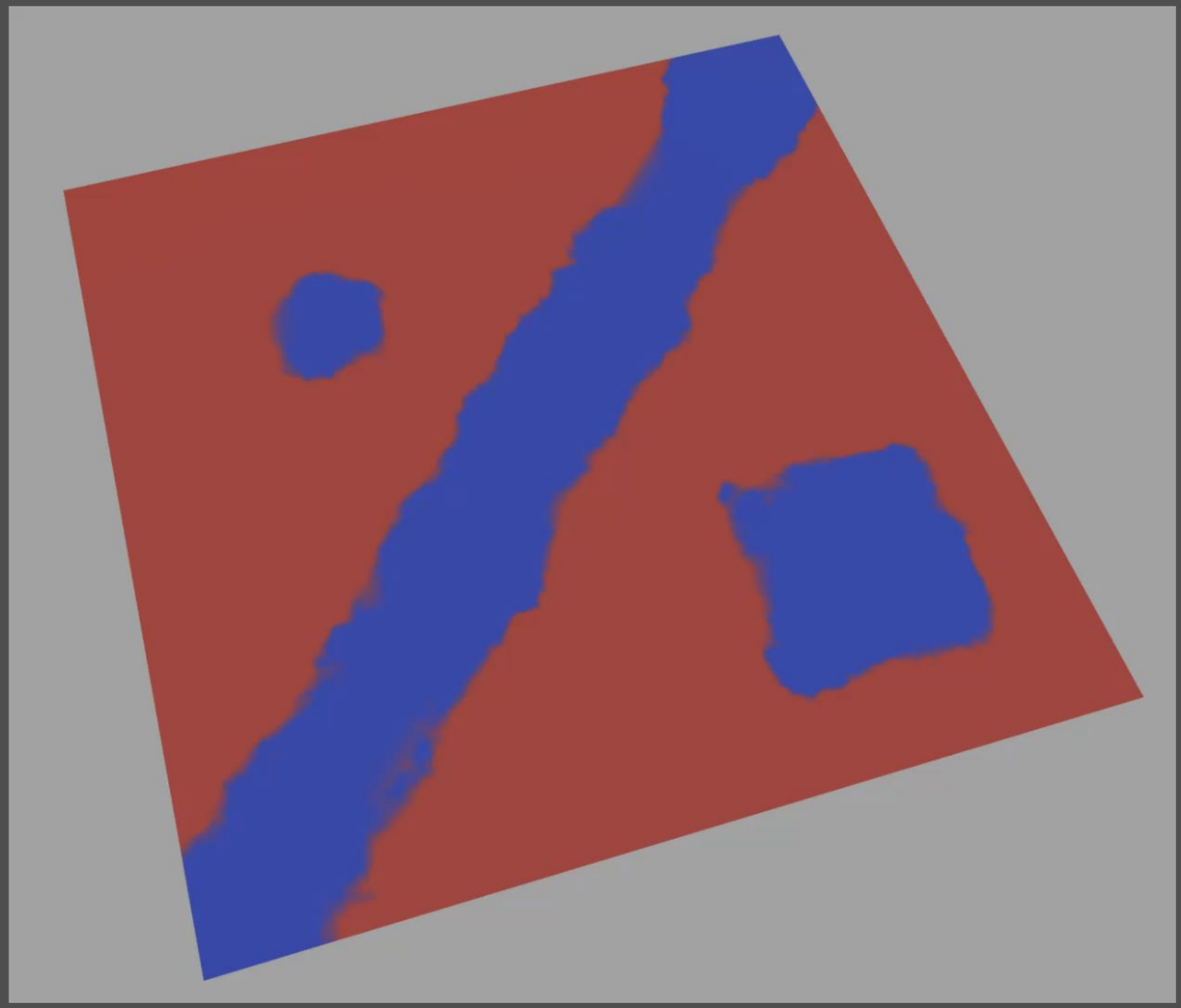
Threshold blend mask





Histogram select

- ‘Histogram select’ = select a specific a position in the histogram created by the previous two blends



Layer stack

- Base layer

Layer stack

- Base layer
- Plastic wear

Layer stack

- Base layer
- Plastic wear
- Rust

Layer stack

- Base layer
- Plastic wear
- Rust
- Scratches and dents

Layer stack

- Base layer
- Plastic wear
- Rust
- Scratches and dents
- Peeling paint

Layer stack

- Base layer
- Plastic wear
- Rust
- Scratches and dents
- Peeling paint
- Rust bleed

Layer stack

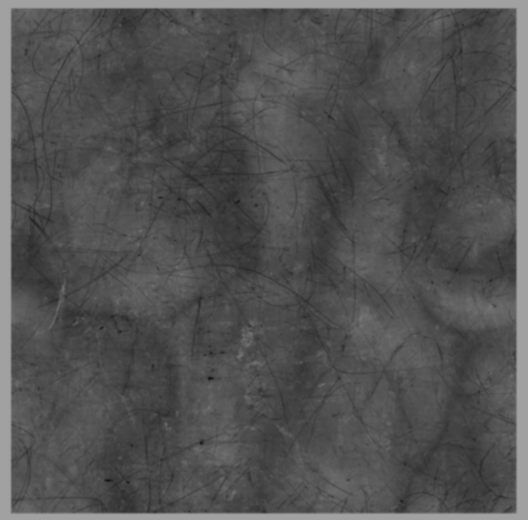
- Base layer
- Plastic wear
- Rust
- Scratches and dents
- Peeling paint
- Rust bleed
- Overall dirt

Modular textures

- Modular textures, that the shader can use for many inputs
- These three textures do a lot of work

High frequency and noisey

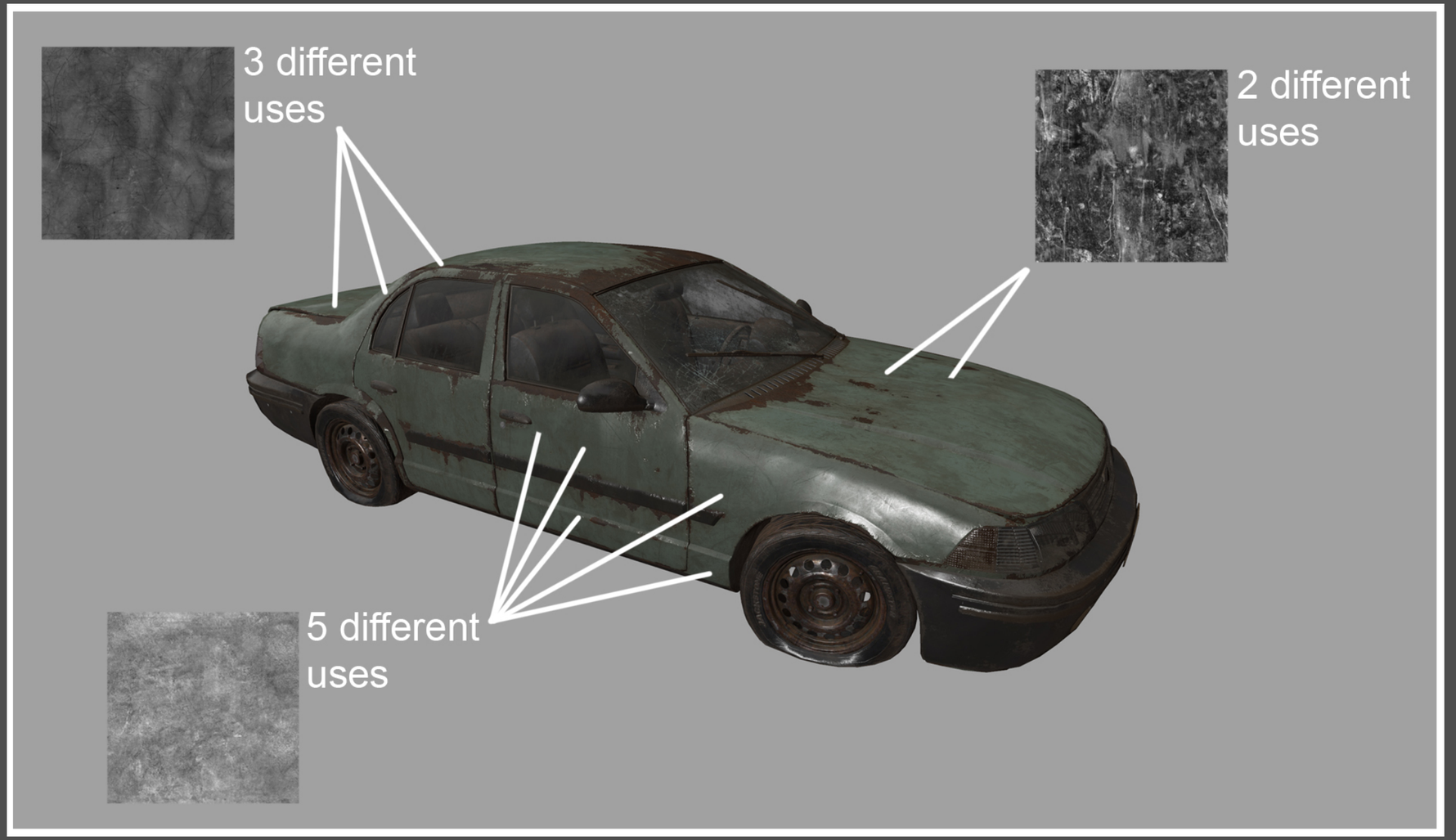
Grunge with good mix of drips, blobs and stains



fine scratches mixed with larger value shifts

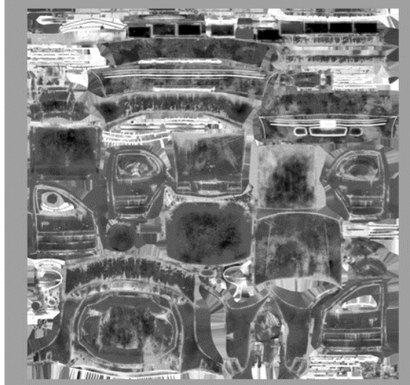
Modular textures

- BRDF textures stripped and replaced with simple colors – given variation via generic threshold textures
- These textures have multiple uses across all layers



Wear mask + threshold

Generic threshold map is also used with the wear mask to add detail to the blend



Threshold blend mask = off

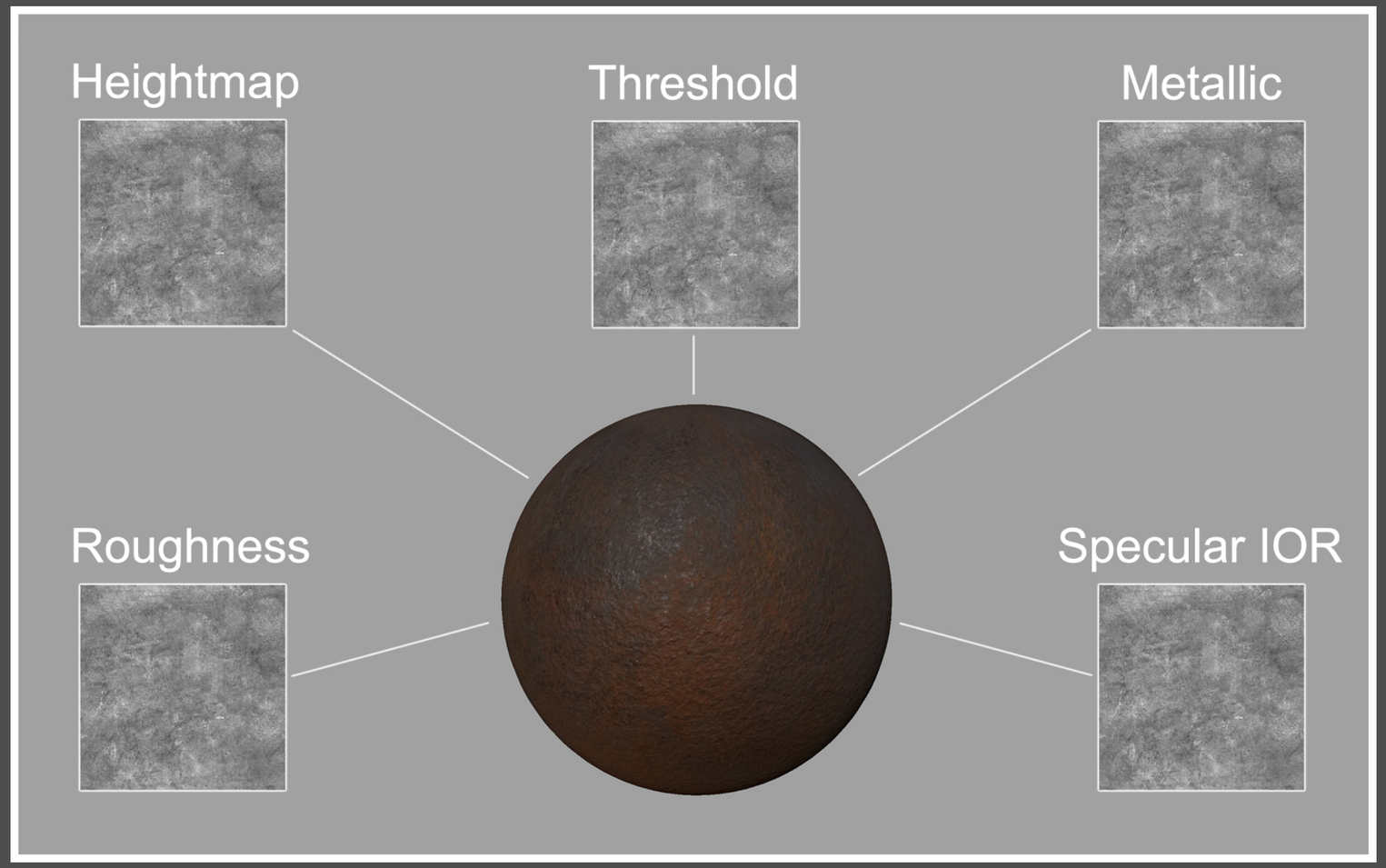
Advanced shader features

- Border blend feature adds detail to the blend
- Leveraging shader features creates detail without the need for additional textures and these effects are easier to optimize

Border blend feature

Another example layer - rust

- Rust works in a similar way
- All layers use this modular texture approach, sharing and re-using where possible



Levels adjustments

- Levels adjustment within shader allows me to fine tune the values the generic threshold texture presents to achieve many different looks

Tiling layer details

- Having layered details that can be tiled independently, means larger vehicles no longer need more textures to look high resolution

Final example layer

- Rust bleed layer is a good example of all three blend methods

BRDF values

- Basic BRDF values (no textures) for the rust bleed effect

Blend against previous layers

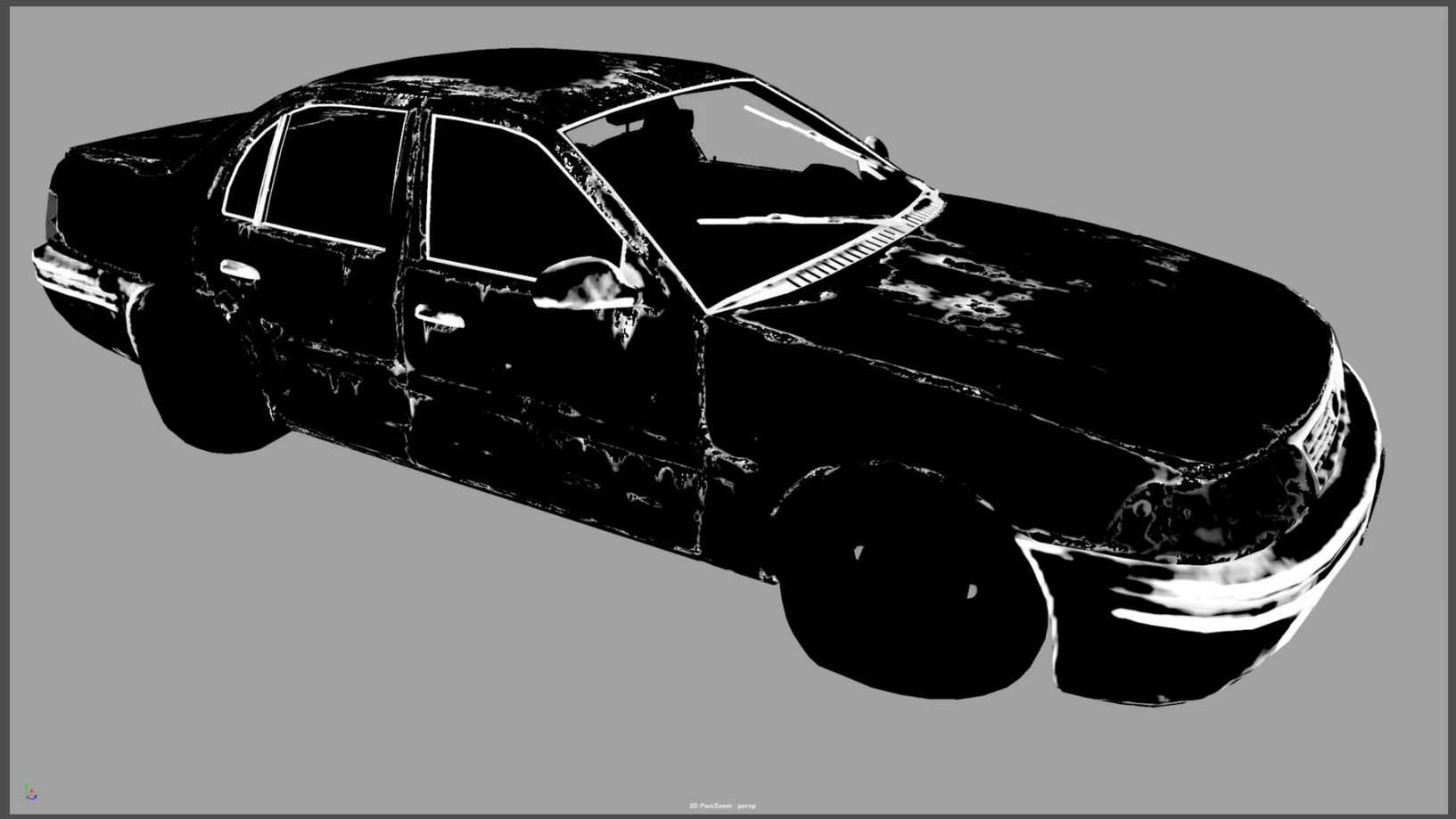
- No longer draws on rust, but still needs more instructions

Wear mask

- The wear mask has the rust bleed drawing in exactly the same position as the basic rust by default

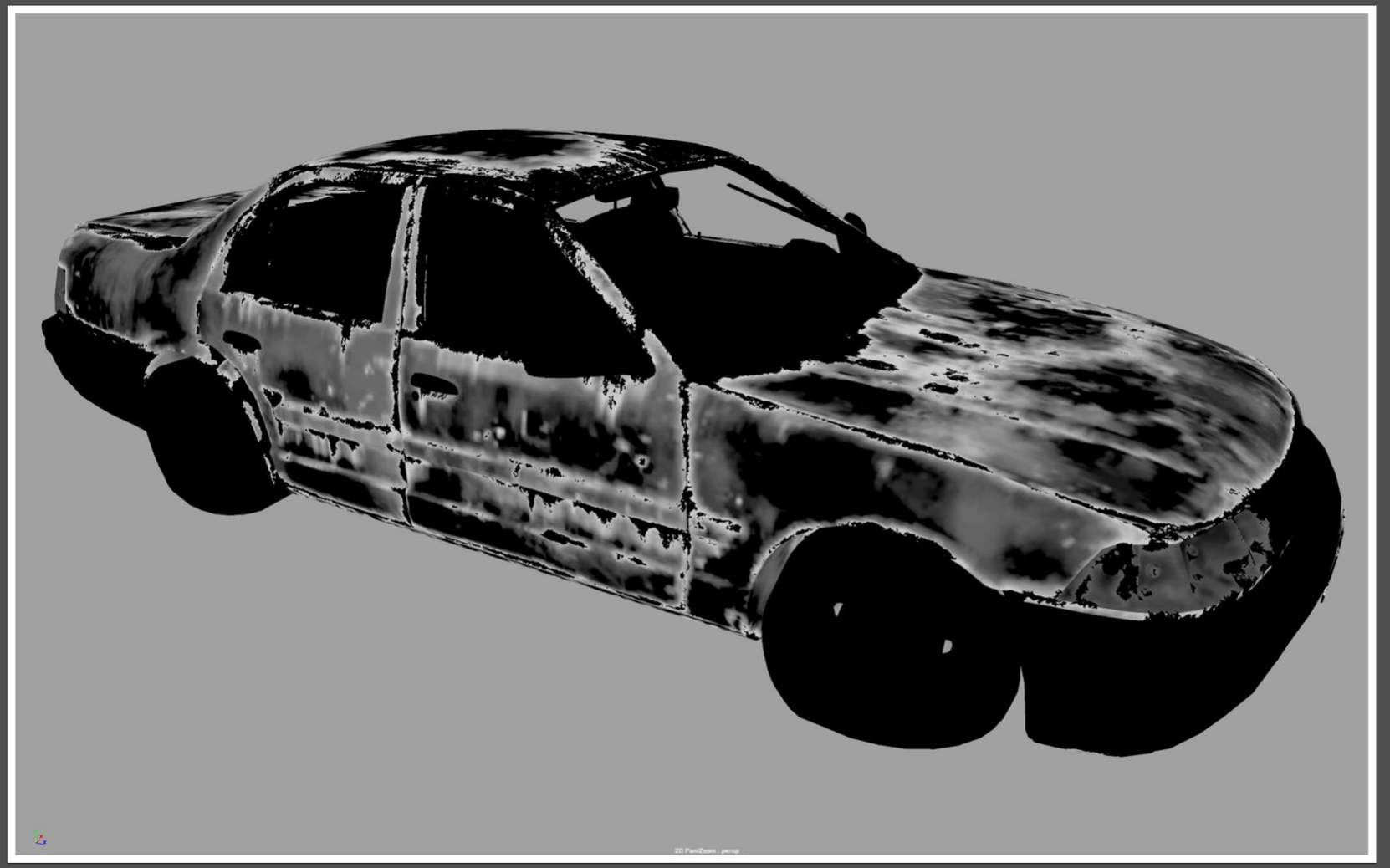
Histogram select

- Using histogram select I can ‘push’ the rust bleed effect out so that it surrounds the rust with a soft effect



Threshold

- The wear mask has the rust bleed drawing in exactly the same position as the basic rust by default



Final appearance

- With my final adjustments the subtle rust bleed layer is complete. It’s cheap and adds a nice element of detail

Texture memory comparison

- Let's see how the new pipeline affects texture memory

Single vehicle

- Around 33 MB for one vehicle on the new pipeline
- Because we have more details and more textures overall our memory for one vehicle has increased.
- Loading an additional Uncharted 4 van cost 9MB

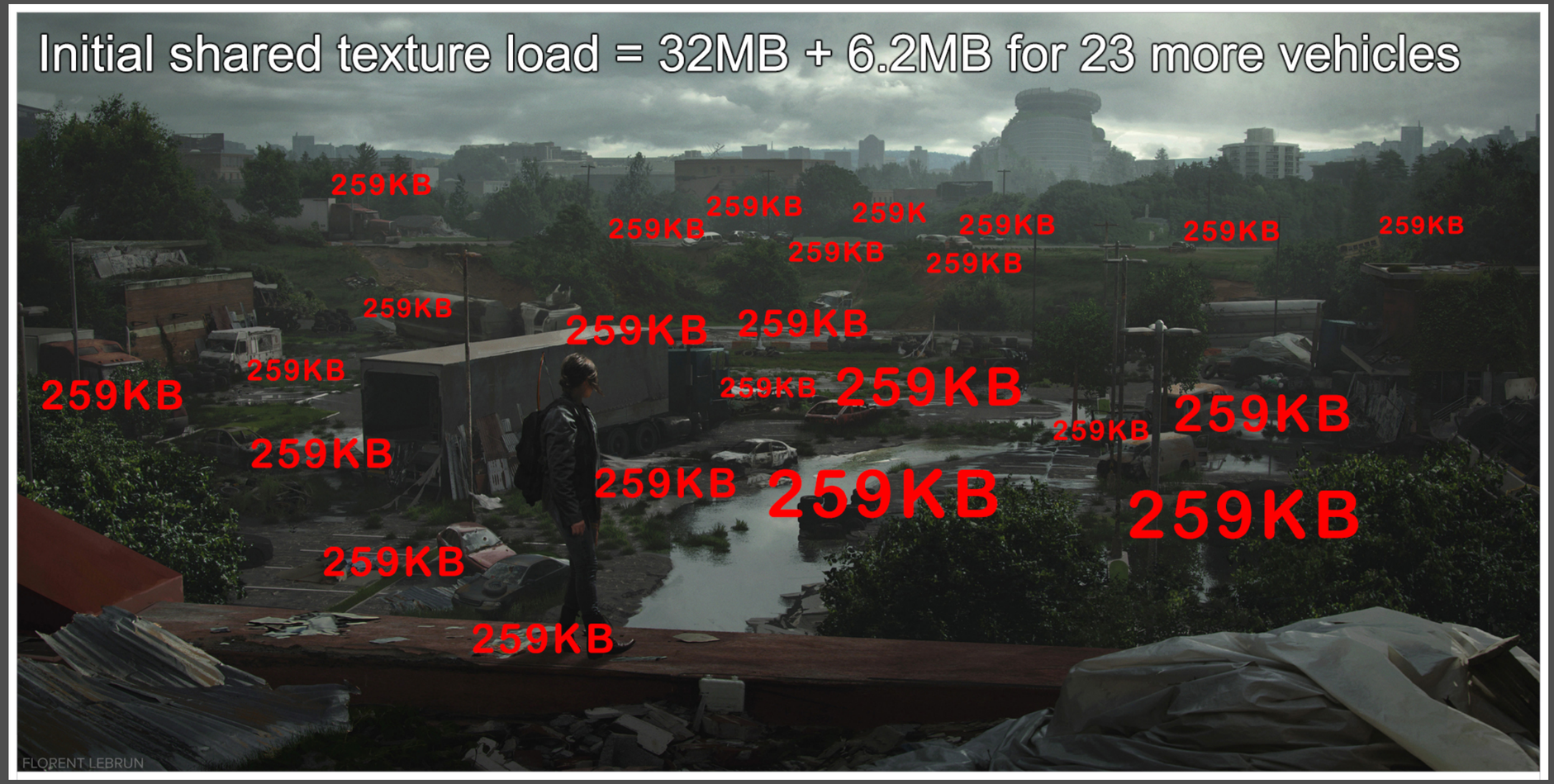
Two vehicles

- Loading an additional TLOU: II van costs just

  259 KB

Entire levels?

- Dramatic reduction in texture memory when adding multiple vehicles to a level



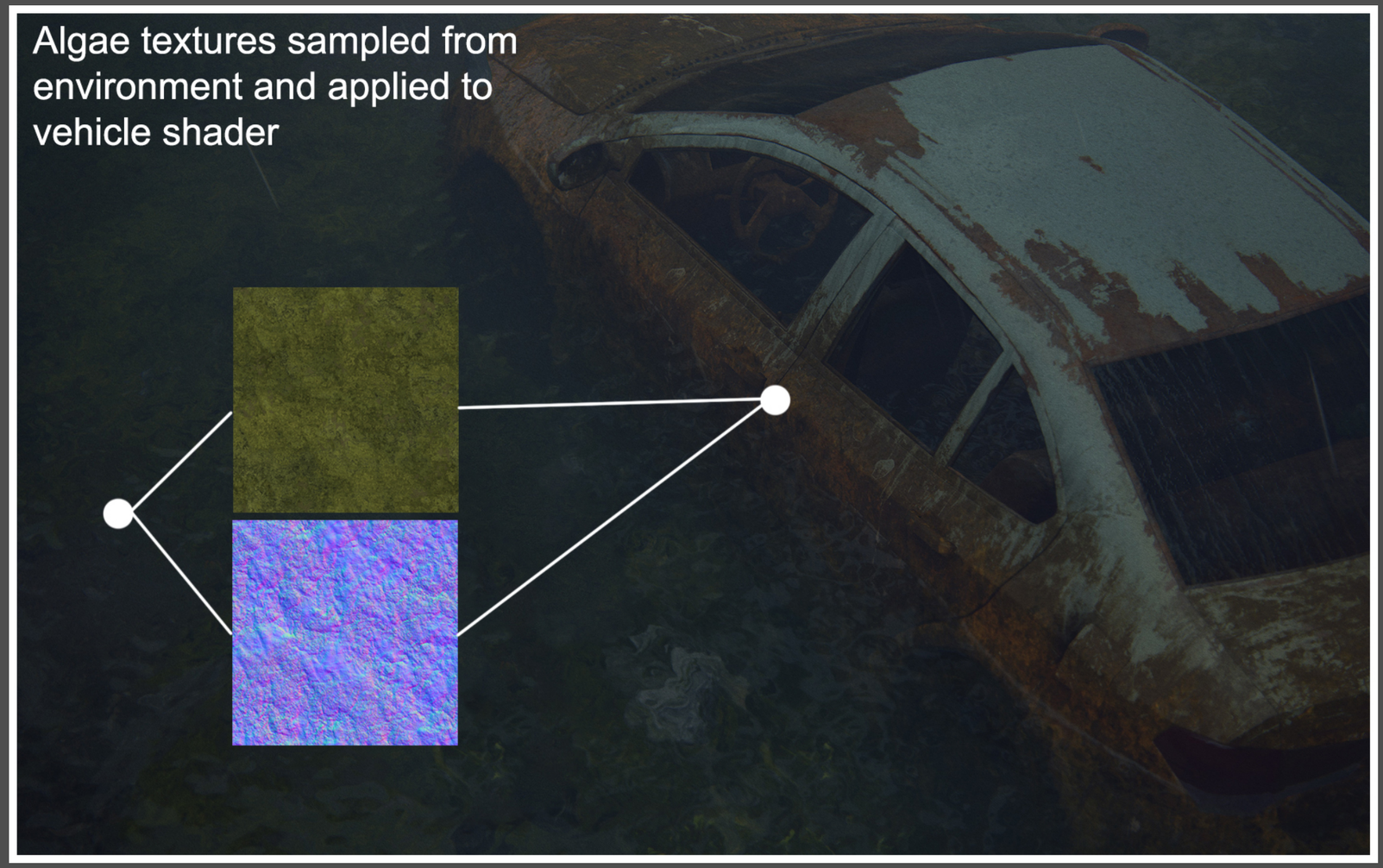
Alternative looks

- All vehicles had multiple shader looks
- Requests for new vehicle types were submitted to myself and Tyler and we created all variations needed

The submerged look

Borrowed textures

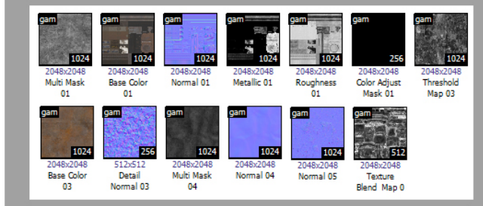
- Making use of already loaded textures to build new shader looks
- This was the case for all of the vehicle looks



Instanced materials

Optimizing the layers

- Added two texture samples (algae) so two are removed (subtle roughness)
- Reducing texture samples in the shader to make it optimized at render time
- Replacing textures, with flat values

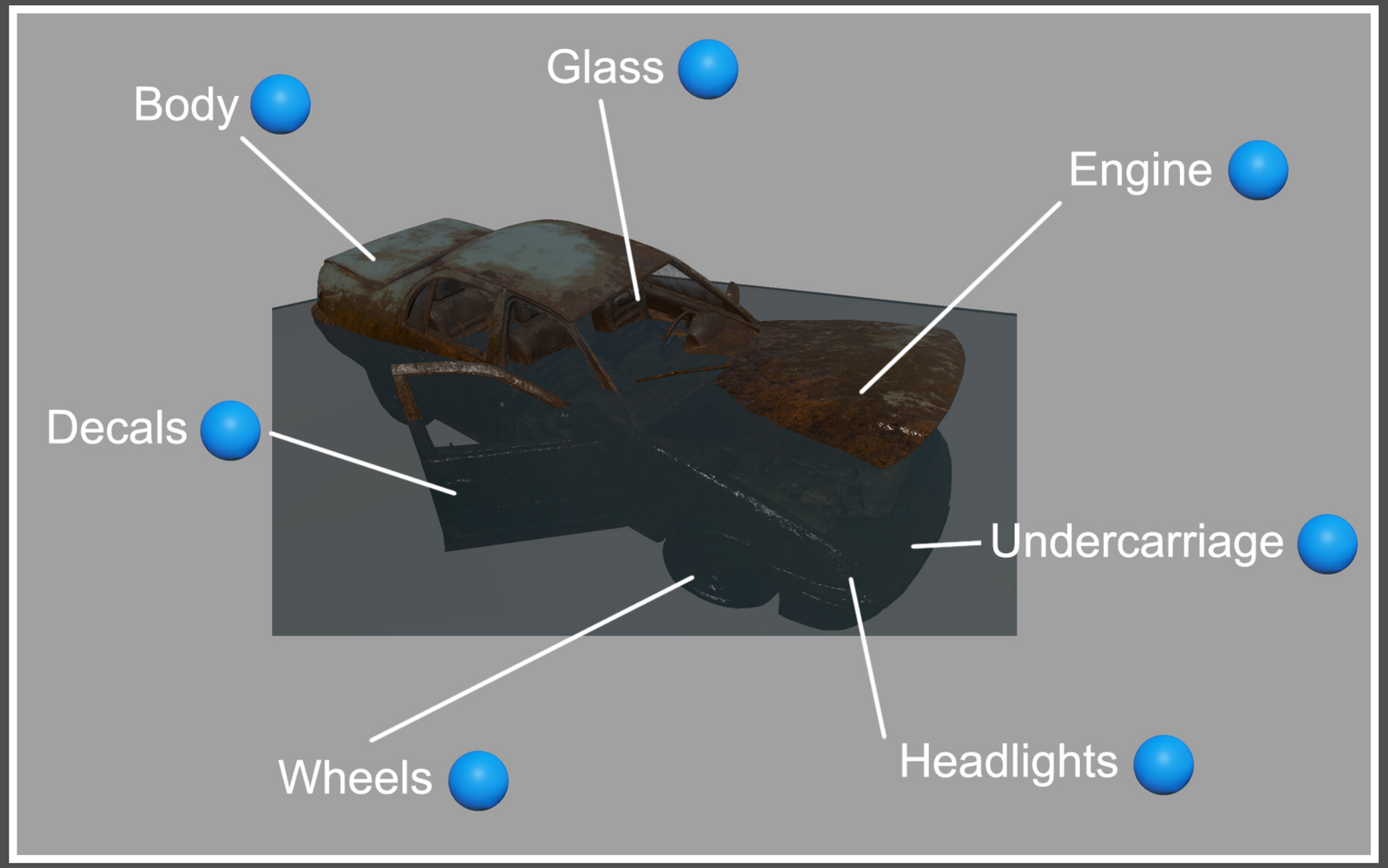


Textures loaded by default look

Textures loaded by submerged look

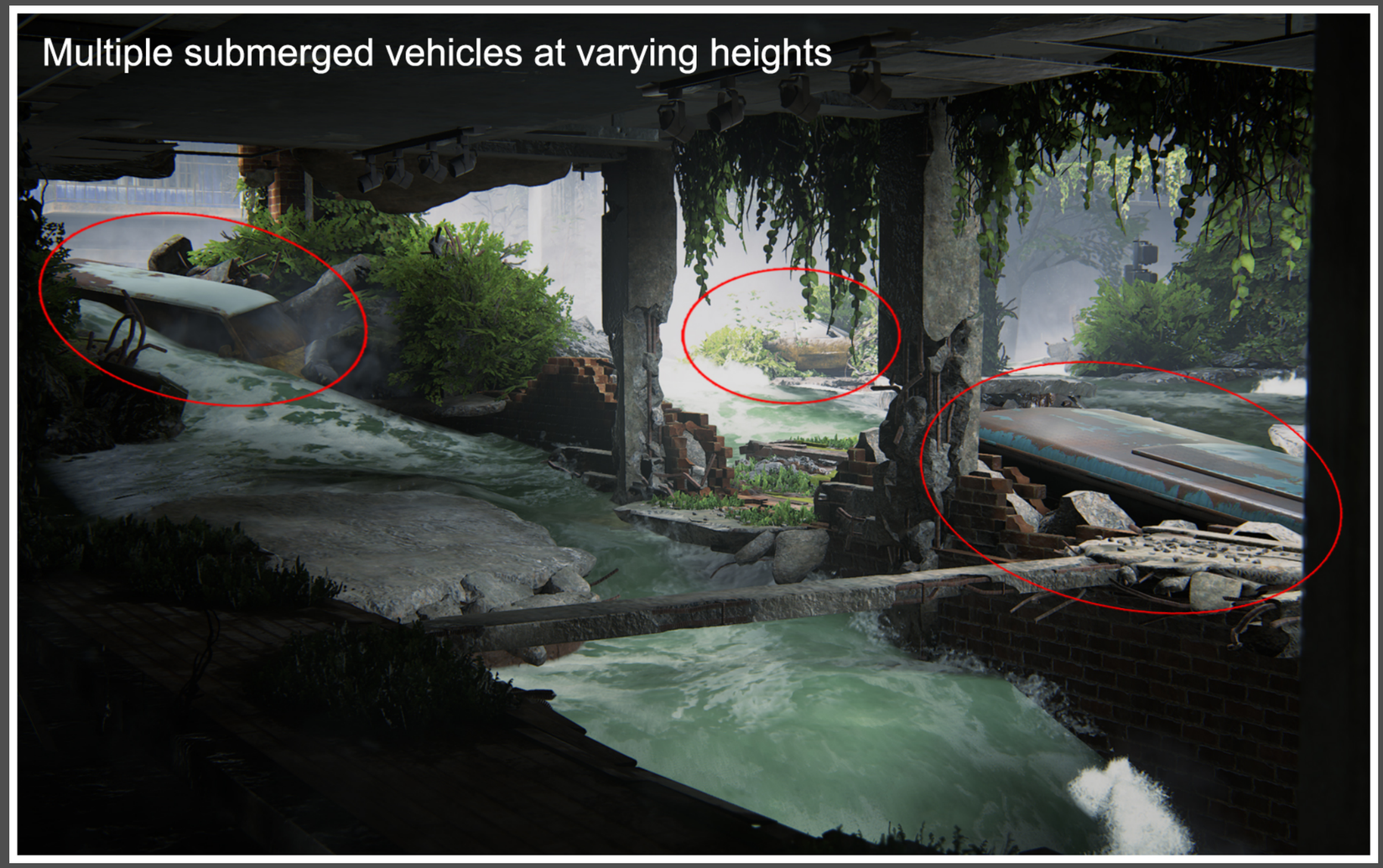
One shader per element

- Each vehicle uses multiple shaders and each shader, has multiple looks



Implementing the blend

- Simple BRDF, much more interesting blend
- Height value drives shader blend
- Old system required height value input per shader



Instance based blends

- World space position and rotation per instance now drives blend
- Allowed us to solve problems like the river rapids sections efficiently

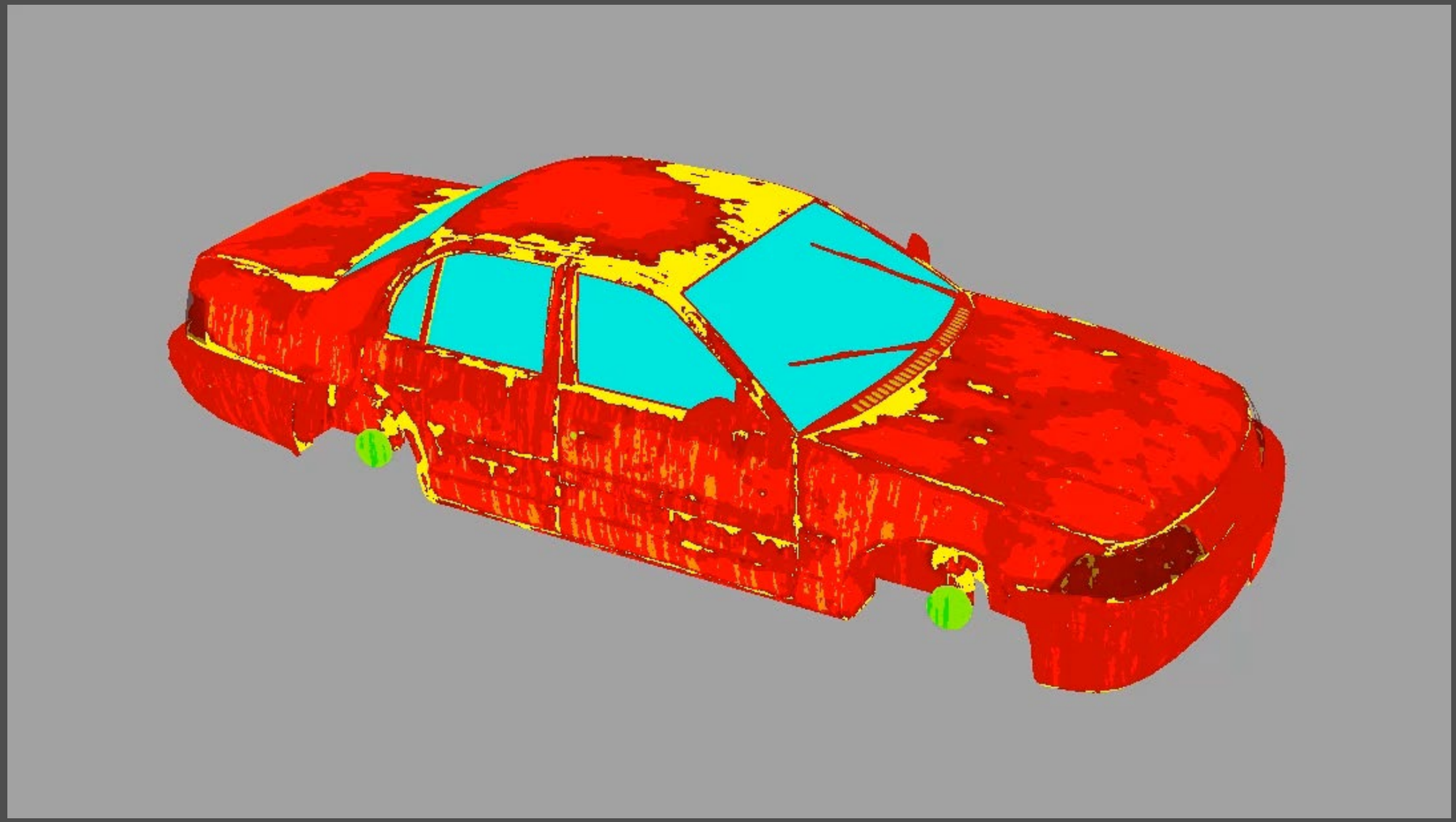
One shader, one instanced vehicle,
but two varying elevation blends

Cool tools

- World space position and rotation values requiring manual entry at first was tricky
- Tool implemented by Dongsub Woo, Steven Tang and Ke Xu

Optimization

- Finally, optimization!
- Layered shaders introduce more texture samples
- Previously mentioned optimizations reduced cost
- Distance based blends made shader much cheaper over distance
- red = expensive, green = cheap



Optimization

- For scenes like this, the optimization really helped
- Shaders automatically becoming cheaper over distance

Final thoughts