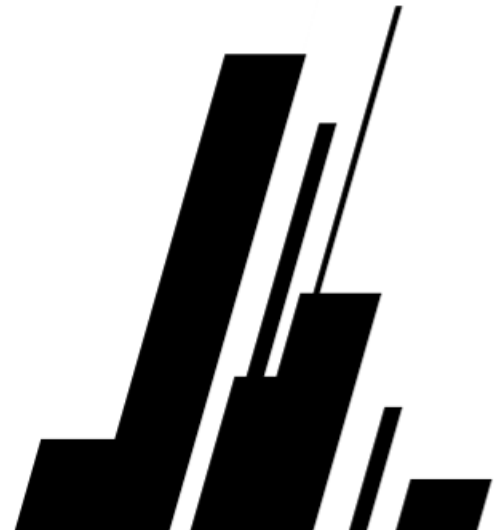
HOW WE RETHOUGHT DEVICE ABSTRACTION



Tamas Rabel
Lead Graphics Programmer

AGENDA

- The abstraction we had
- Problems we faced
- A better way™
- Profit
- Results

AGENDA

- The abstraction we had
- Problems we faced
- A better way™
- Profit
- Results

THE ABSTRACTION WE HAD

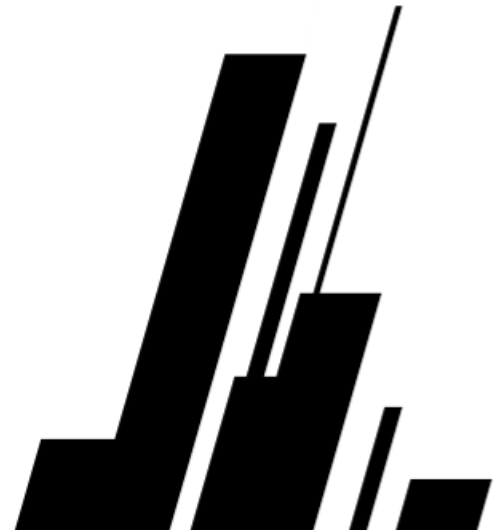
- Started as a thin layer based on DX9
  - Updated to match DX10 later
- Abstracts methods / calls
- Not the rendering process itself

OLD PIPELINE

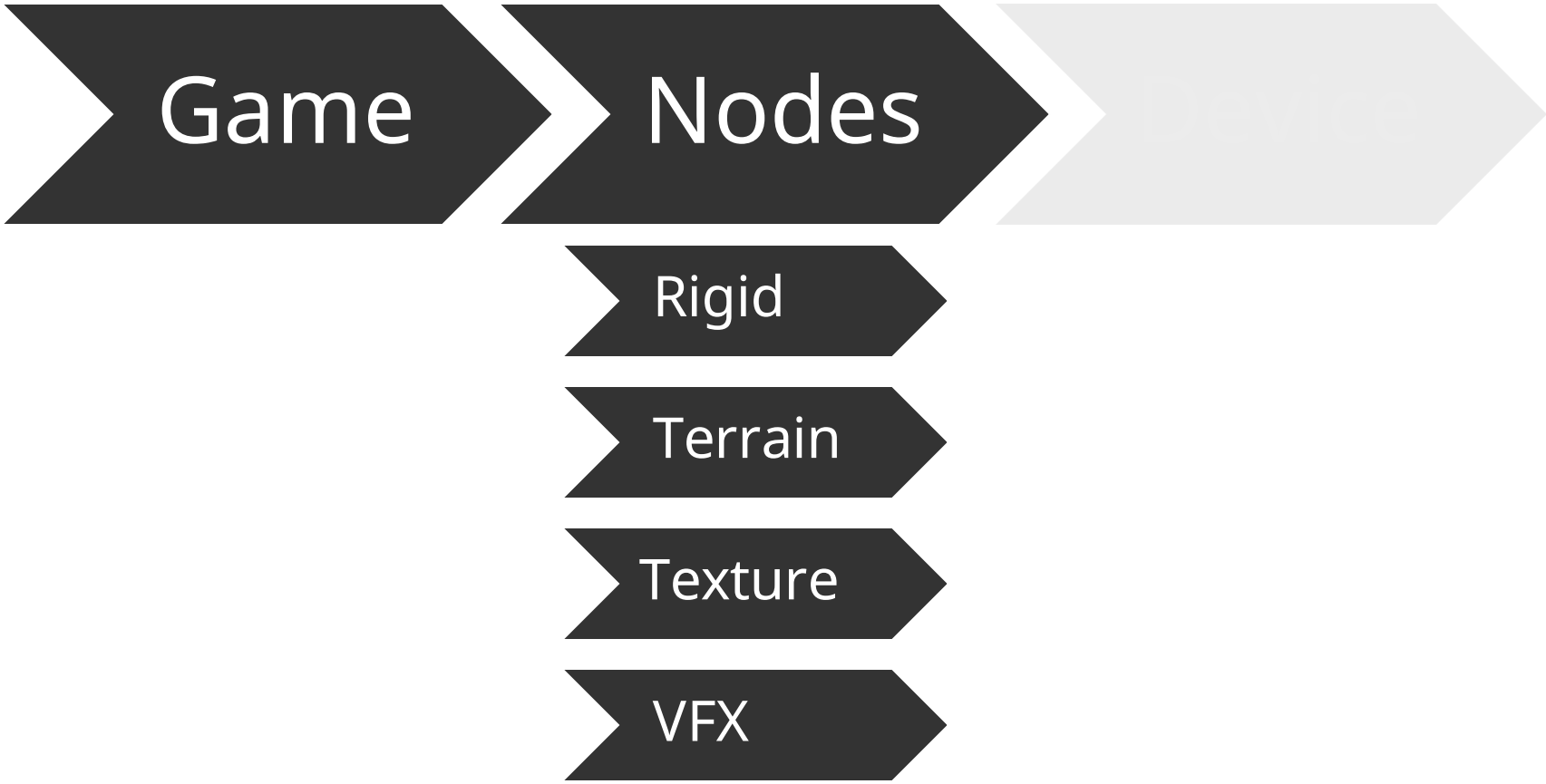
Game

OLD PIPELINE

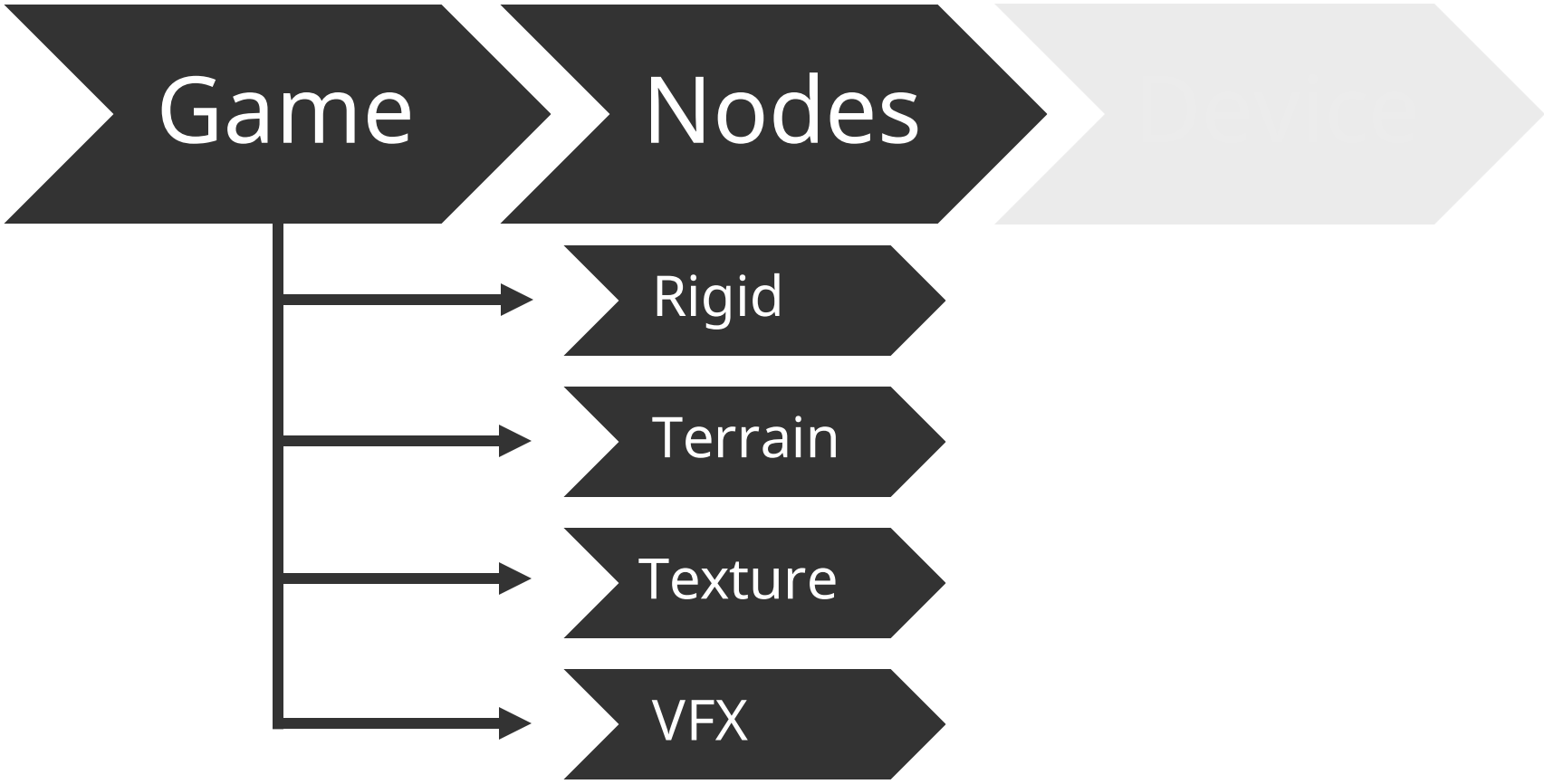
Game → Nodes → [illegible]



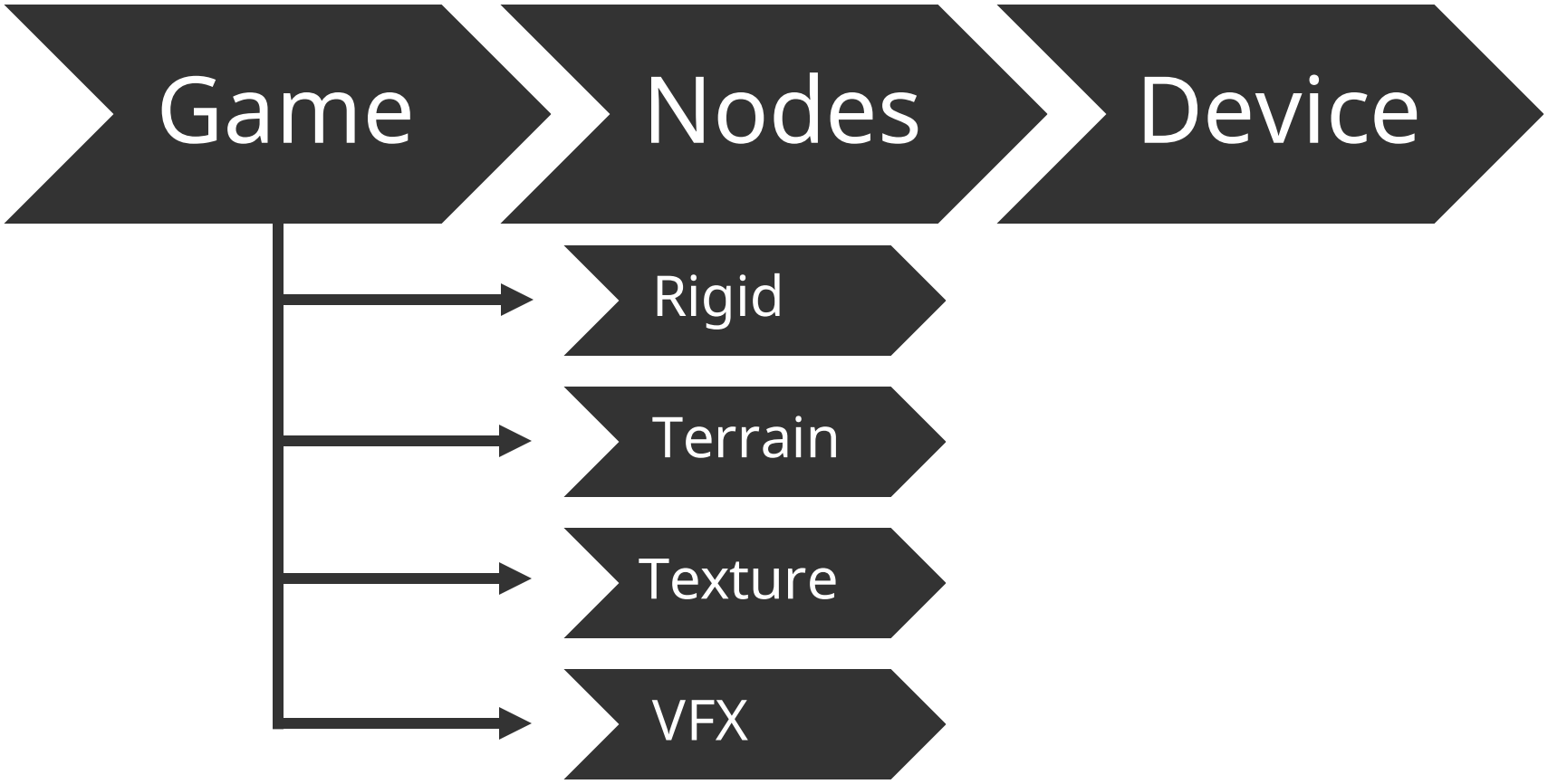
OLD PIPELINE



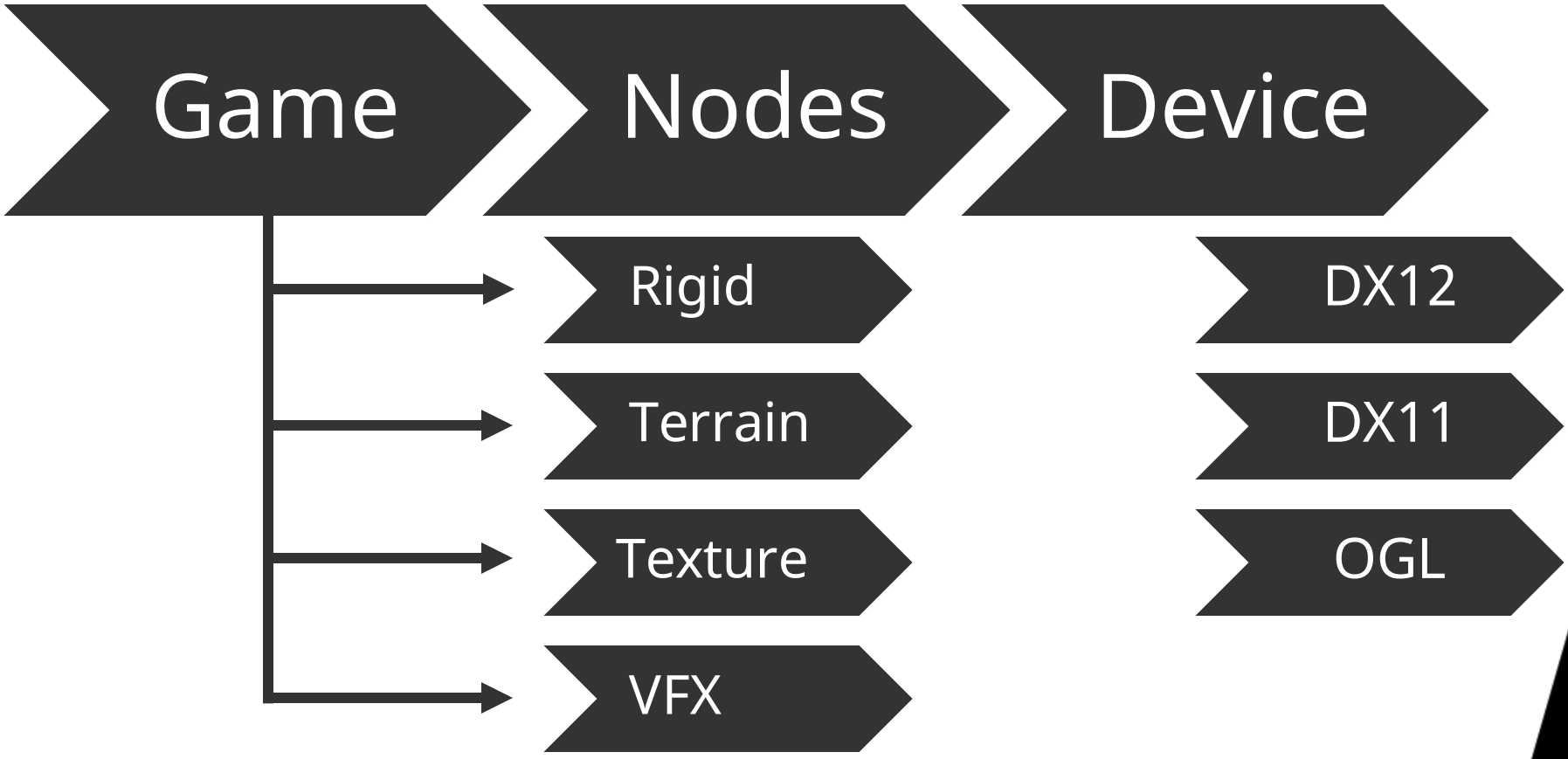
OLD PIPELINE



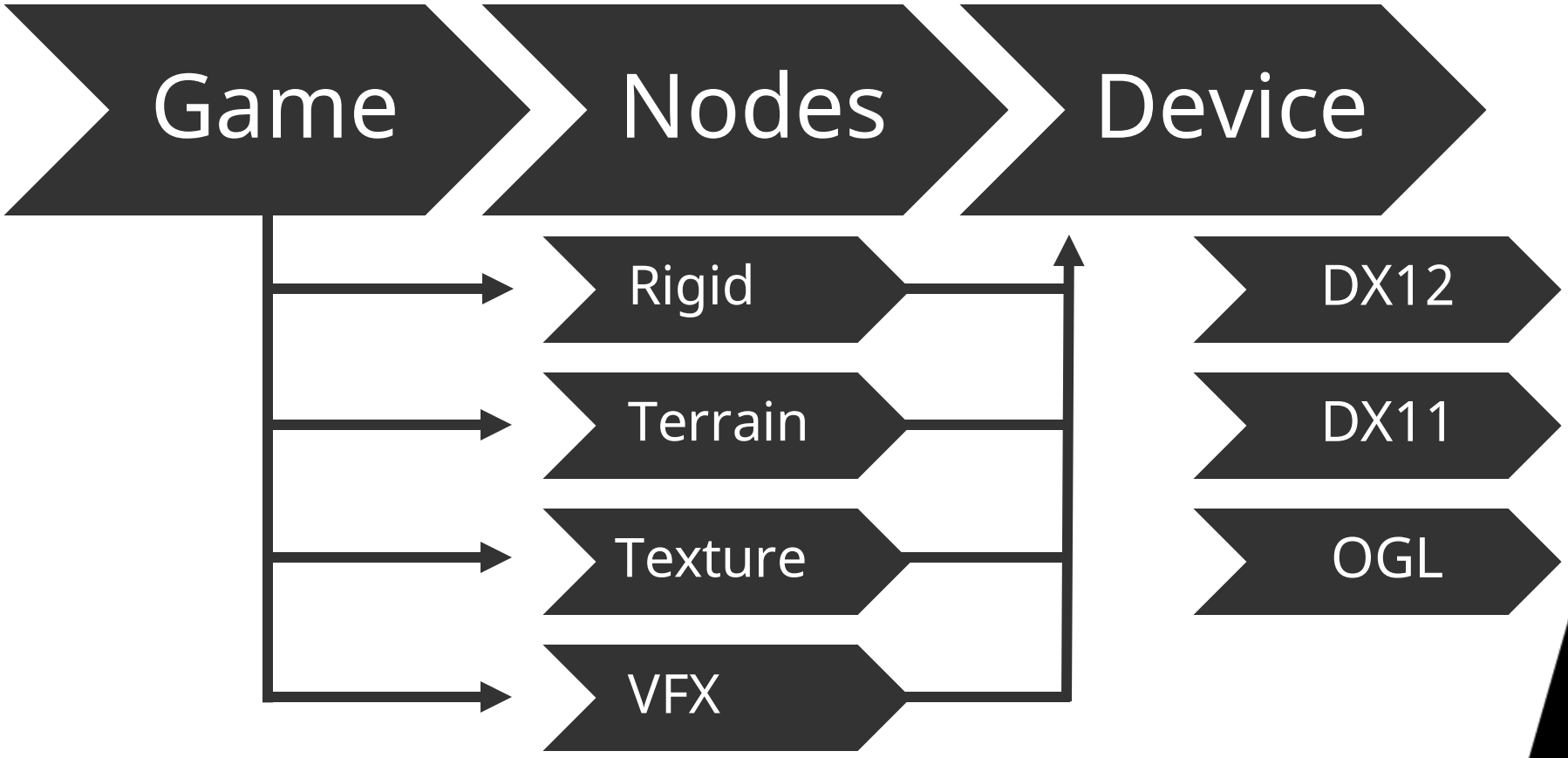
OLD PIPELINE



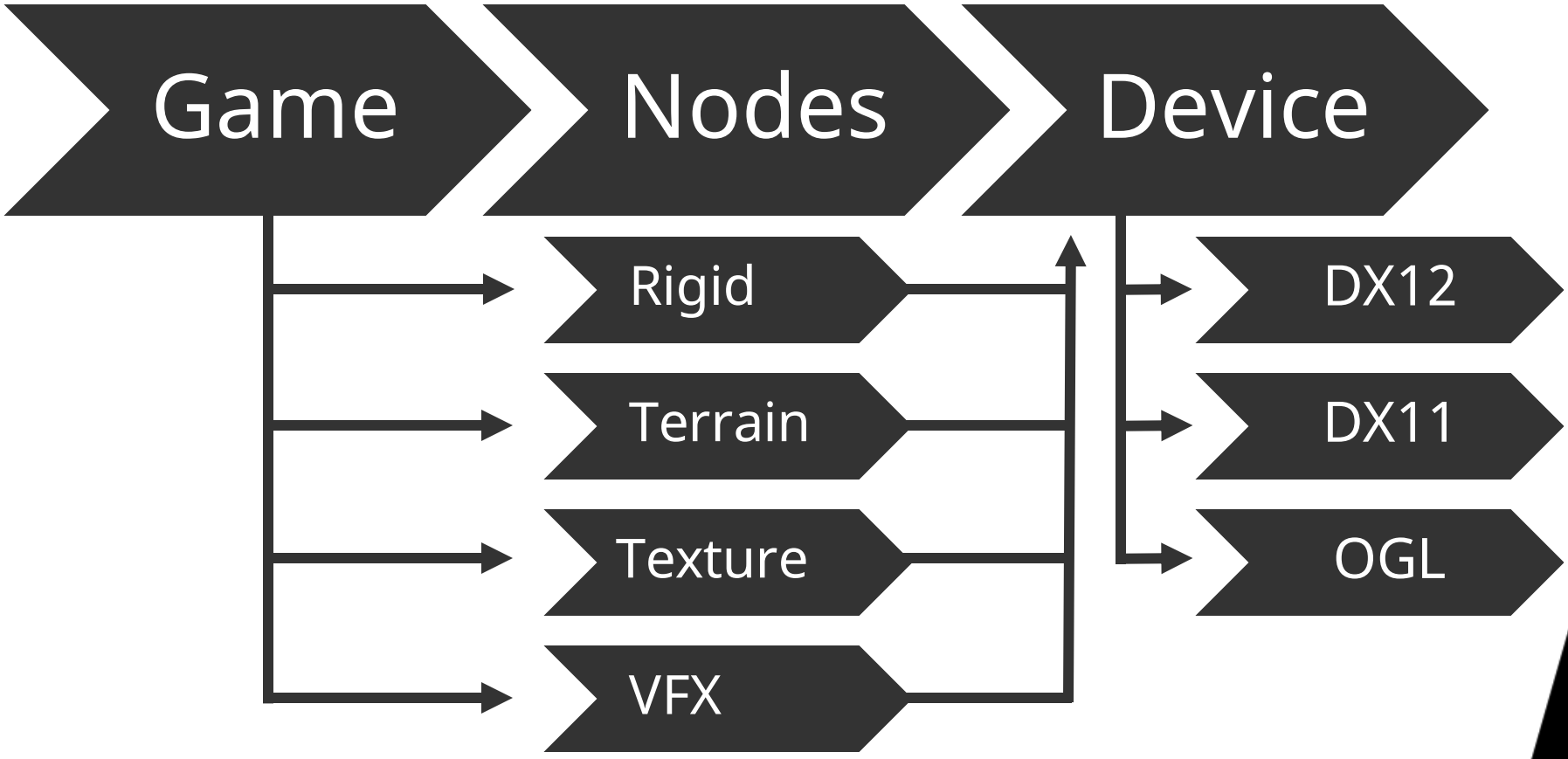
OLD PIPELINE



OLD PIPELINE



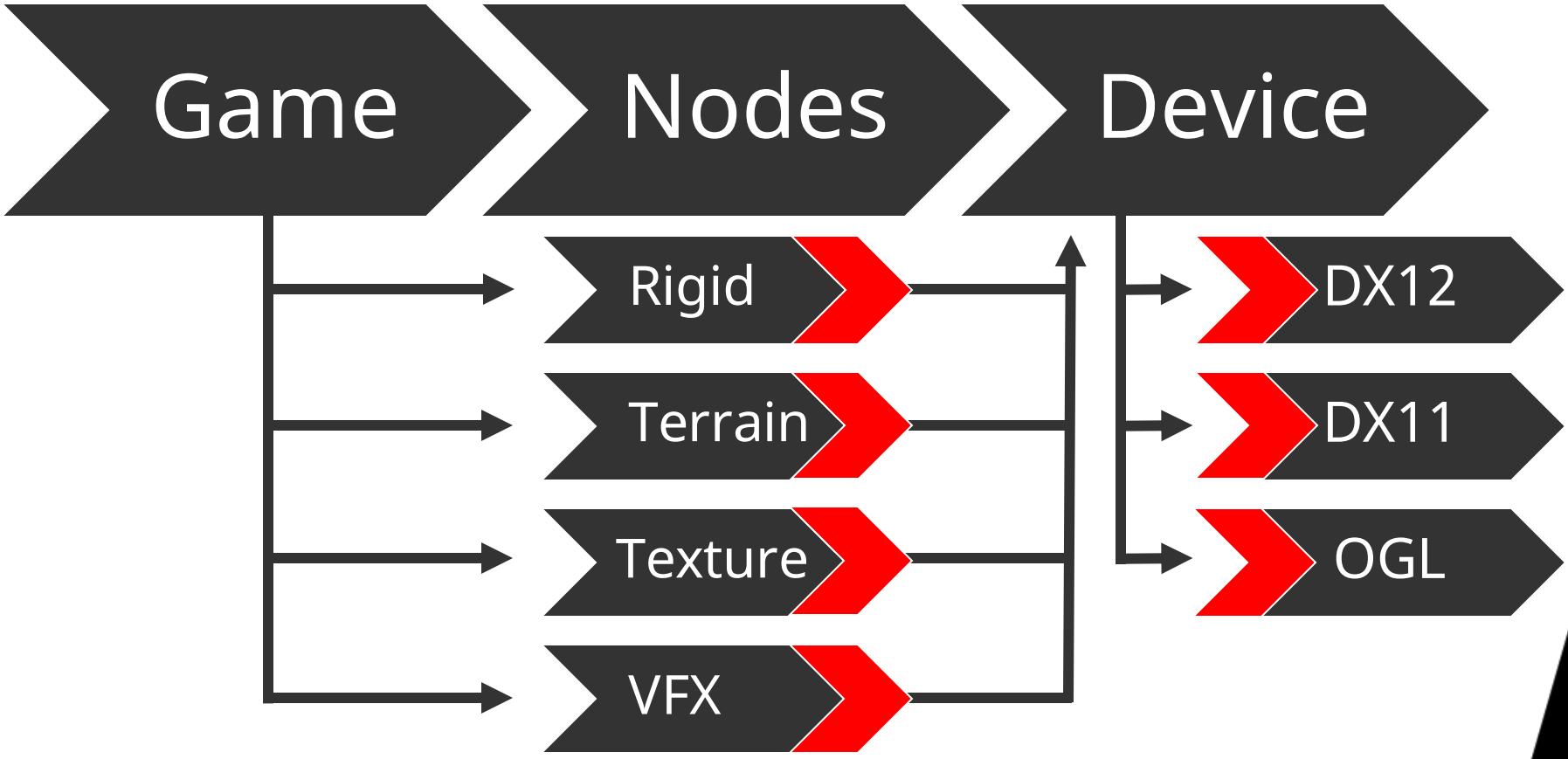
OLD PIPELINE



AGENDA

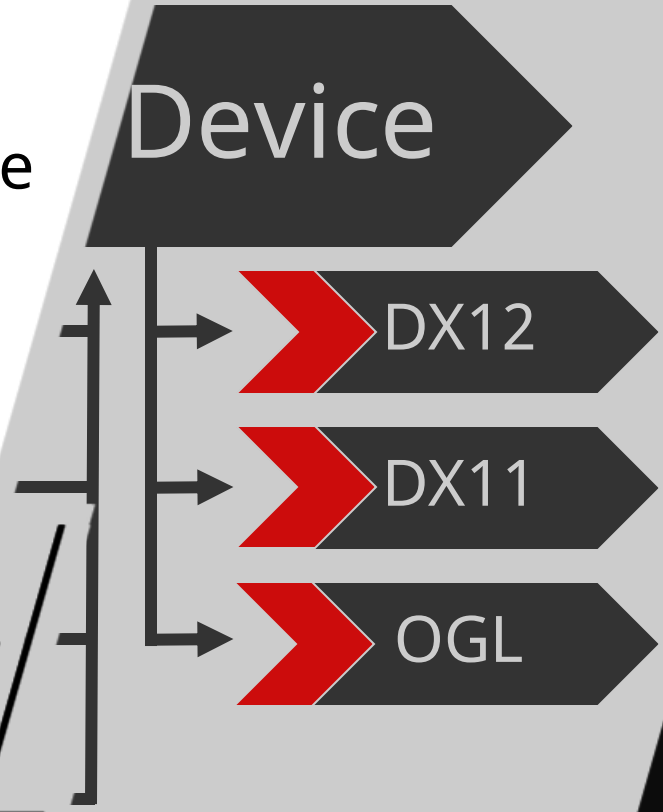
- The abstraction we had
- Problems we faced
- A better way™
- Profit
- Results

OLD PIPELINE

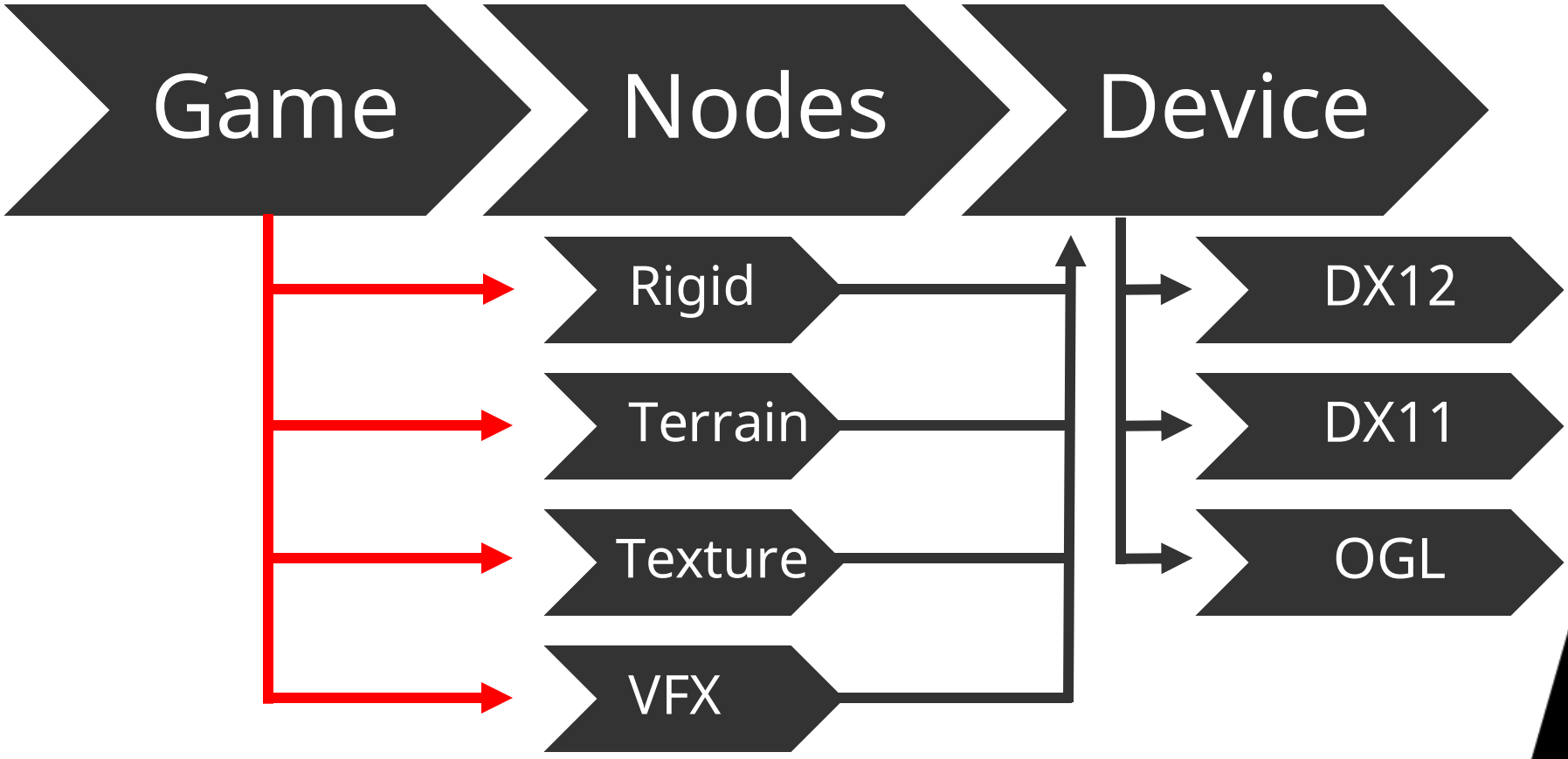


WORK ITEMS

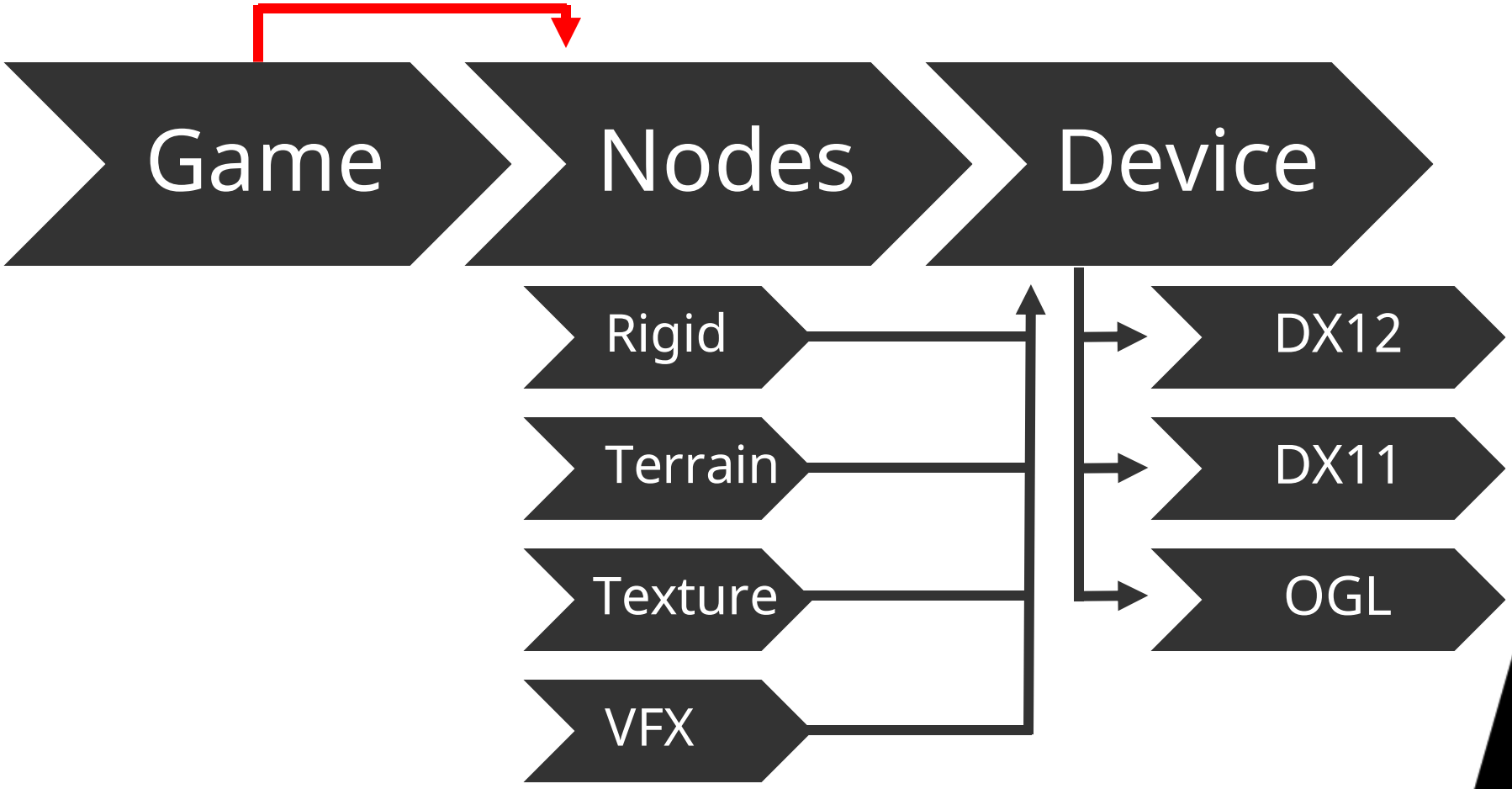
- Minimum data required to describe a single bit of rendering work
  - Inputs
  - Outputs
  - Program(s)
- As generic as possible
- Not necessarily GPU work
- Fully self-contained



OLD PIPELINE

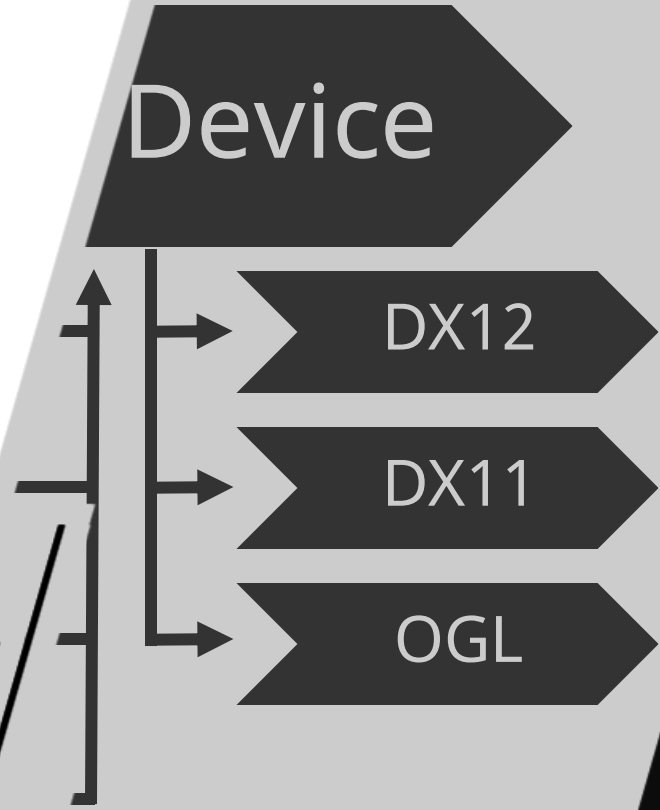


OLD PIPELINE



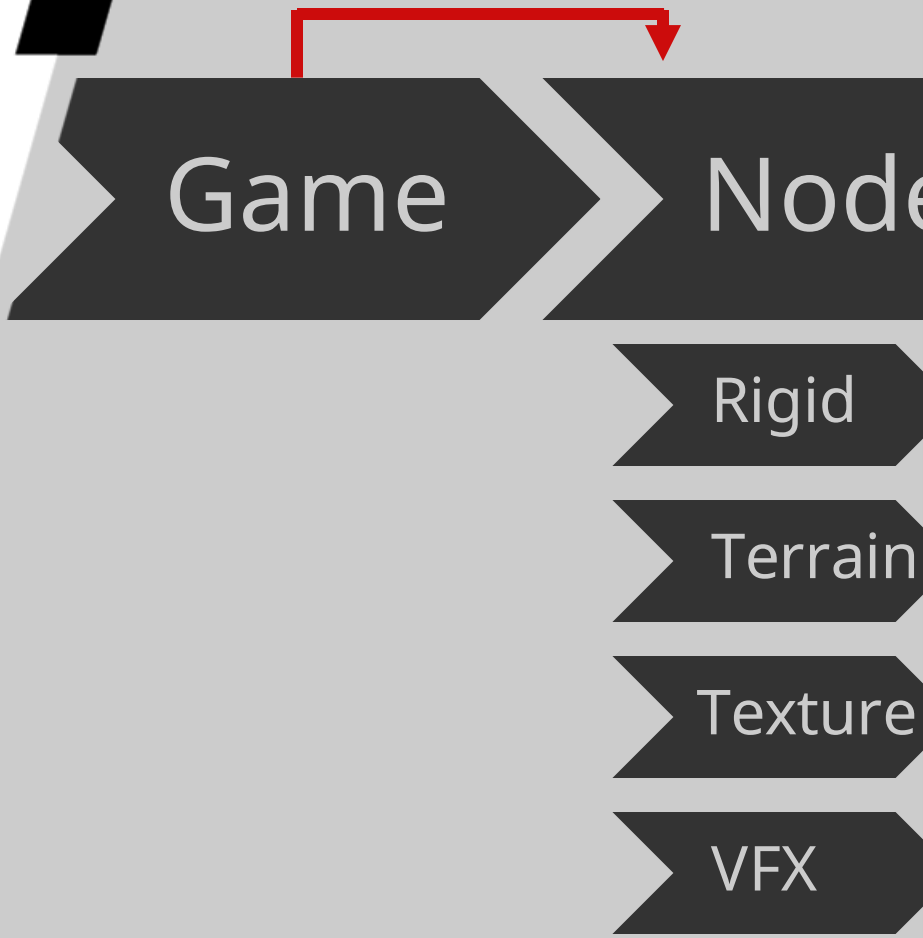
RESOURCES AND INSTANCES

- “Resource” can be anything
  - Black box for the client code
  - Can have different implementations per-platform or even per-run
  - May have instances (see later)
  - Long lifetime
  - Doesn’t necessarily reside in memory for its whole lifetime
- Texture, shader, model, etc.



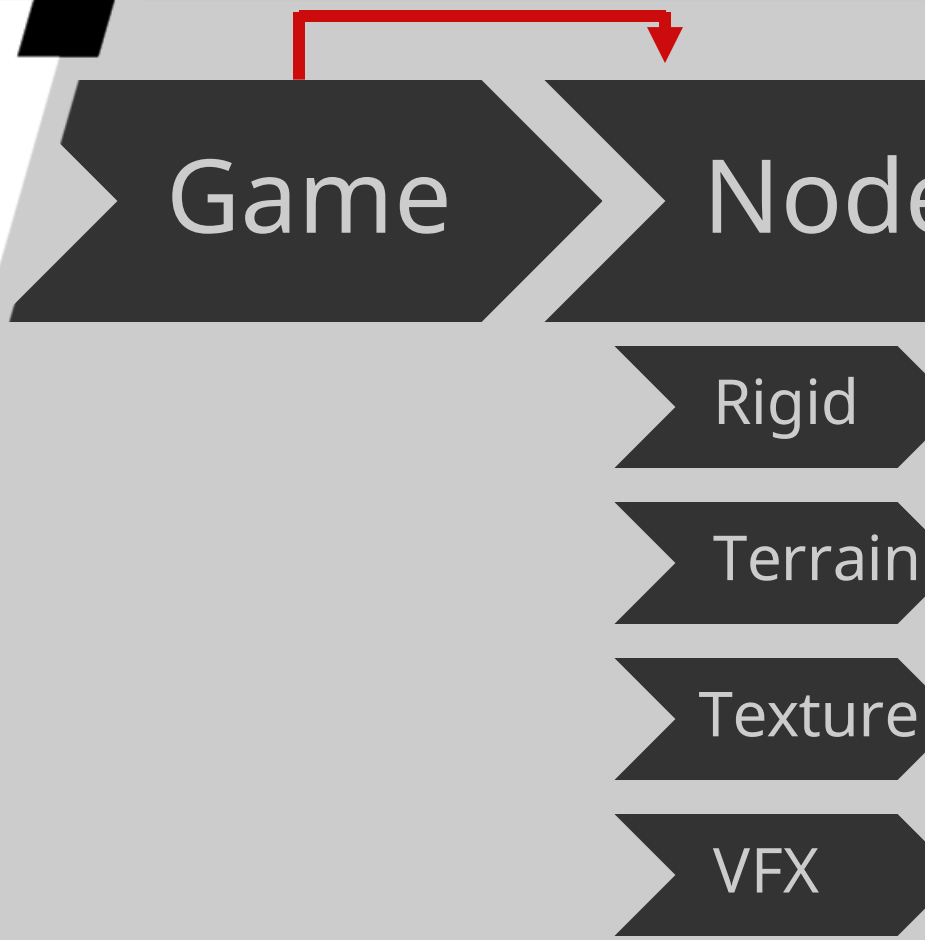
RESOURCES AND INSTANCES (contd.)

- “Instance”
  - of a resource
  - Is transient
  - Black box for the client code
  - Handled by the same code as the resource
- An instance of a model rendered in the current frame for example



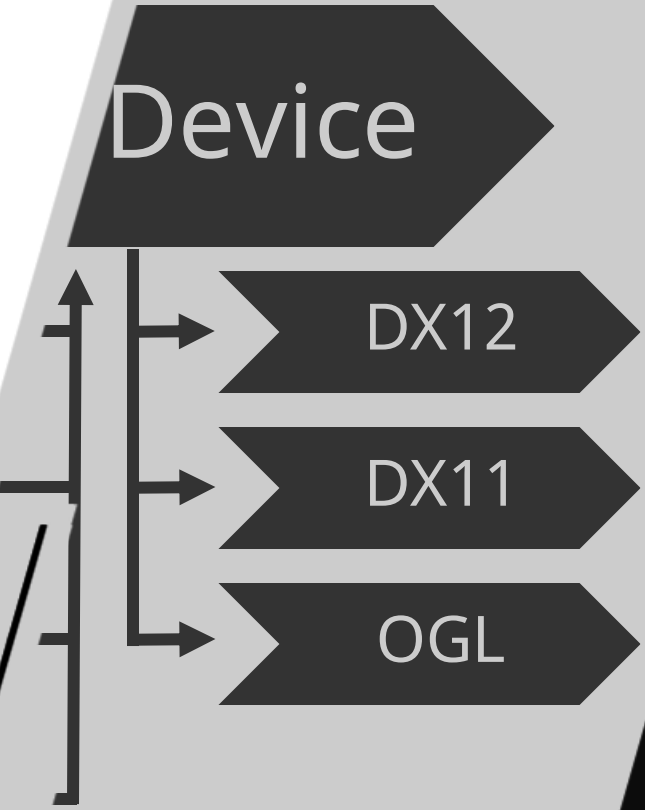
RESOURCES AND INSTANCES (contd.)

- Handles
  - set(colour_handle, colour::red);
  - get(colour_handle)
  - Internally a handle is an offset into instance memory
  - Only way for clients to access resources and instances
  - NOP if the given instance or resource doesn't have the requested parameter
  - Type checked in debug

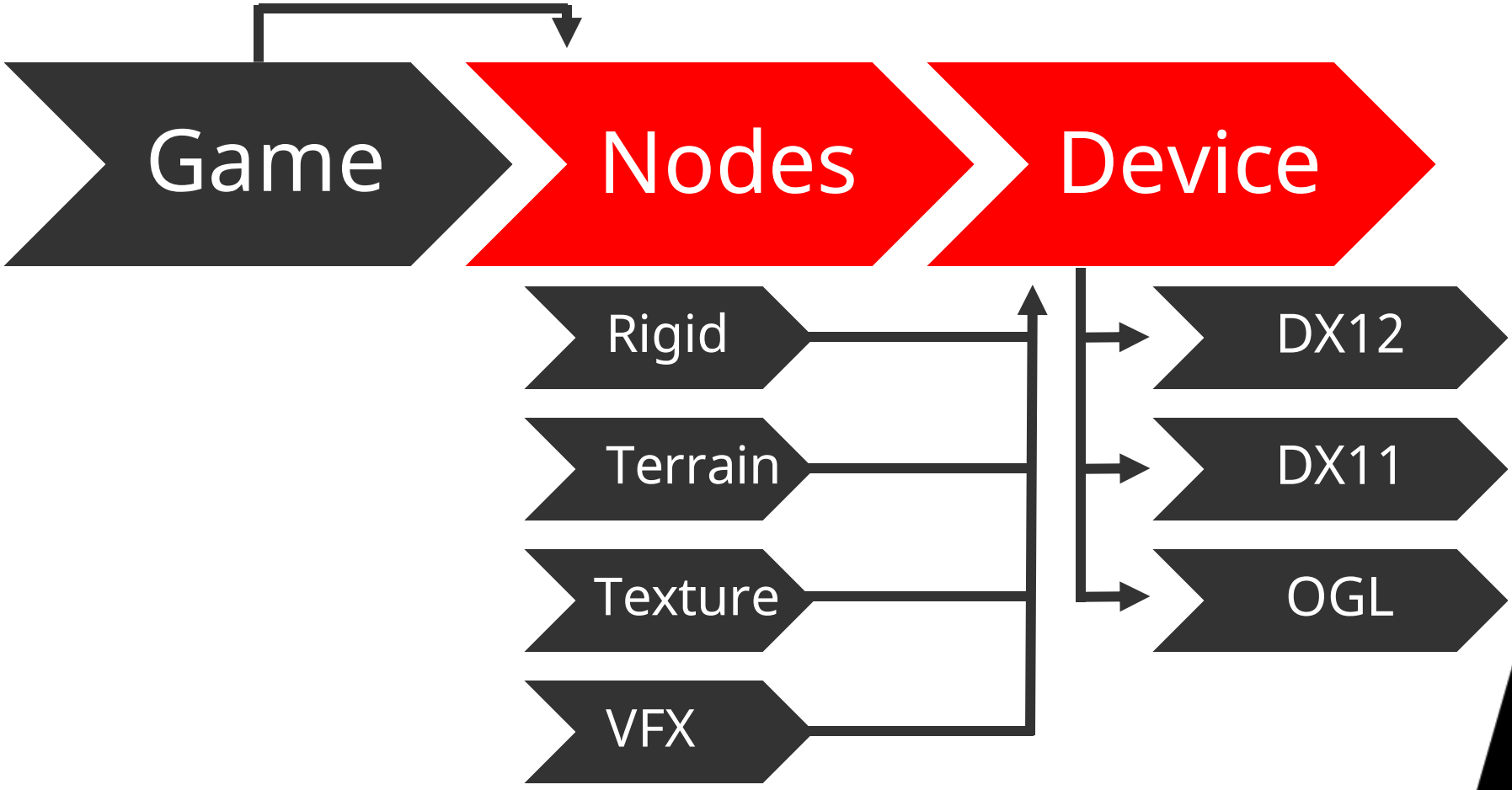


RESOURCES AND INSTANCES

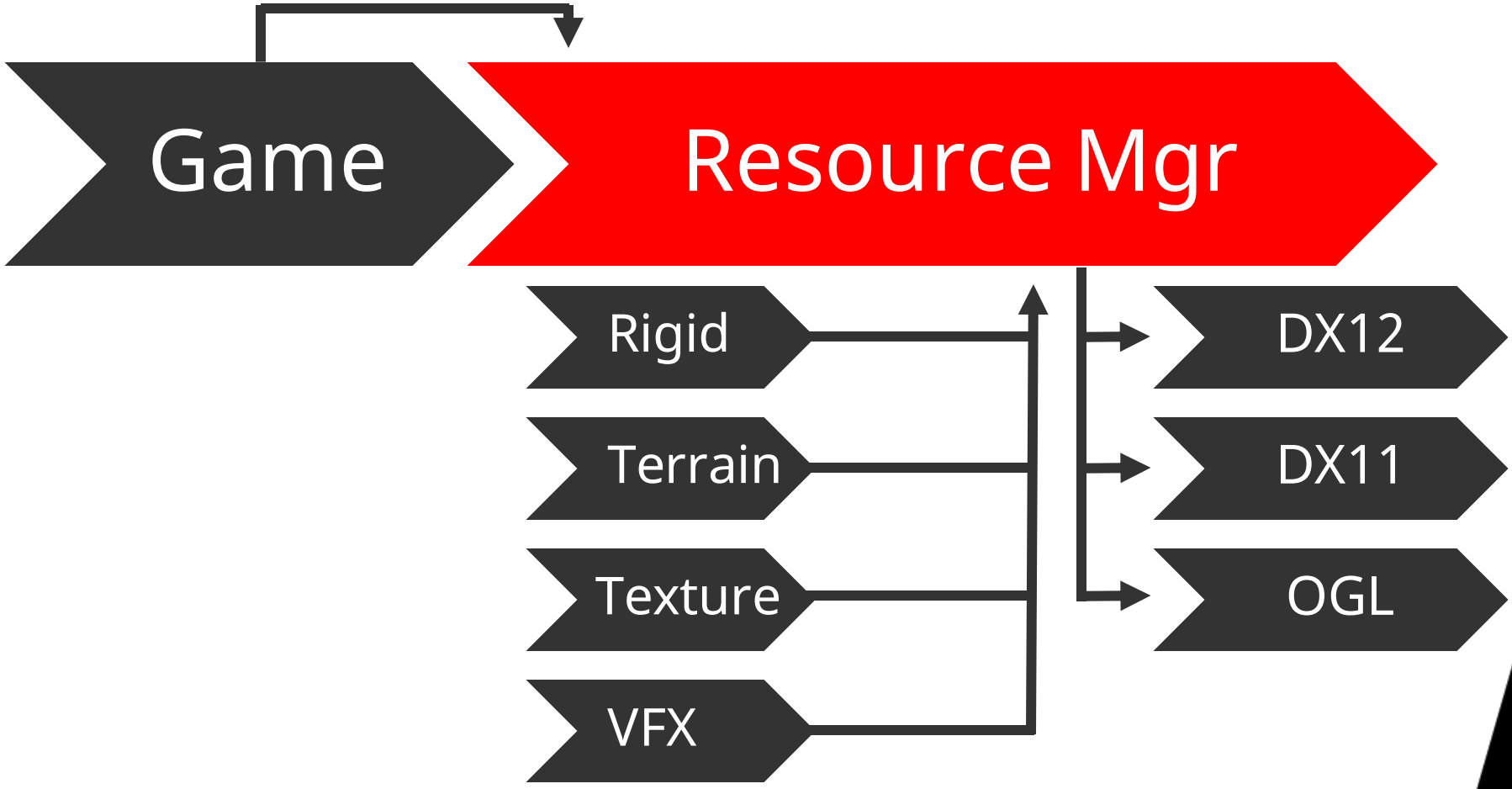
- Handles
  - set(colour_handle, colour::red);
  - get(colour_handle)
  - Internally a handle is an offset into instance memory
  - Only way for clients to access resources and instances
  - NOP if the given instance or resource doesn’t have the requested parameter
  - Type checked in debug



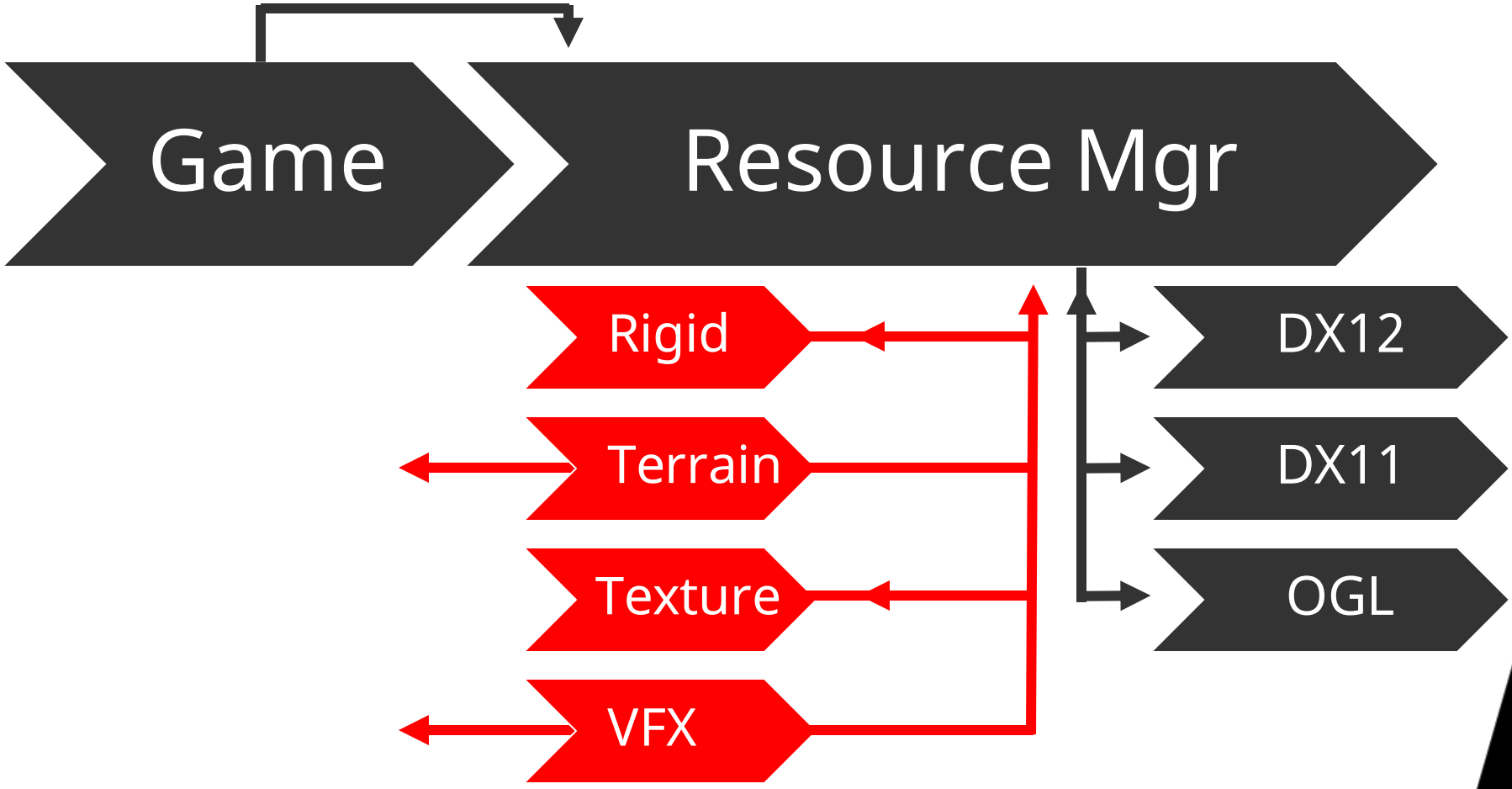
OLD PIPELINE



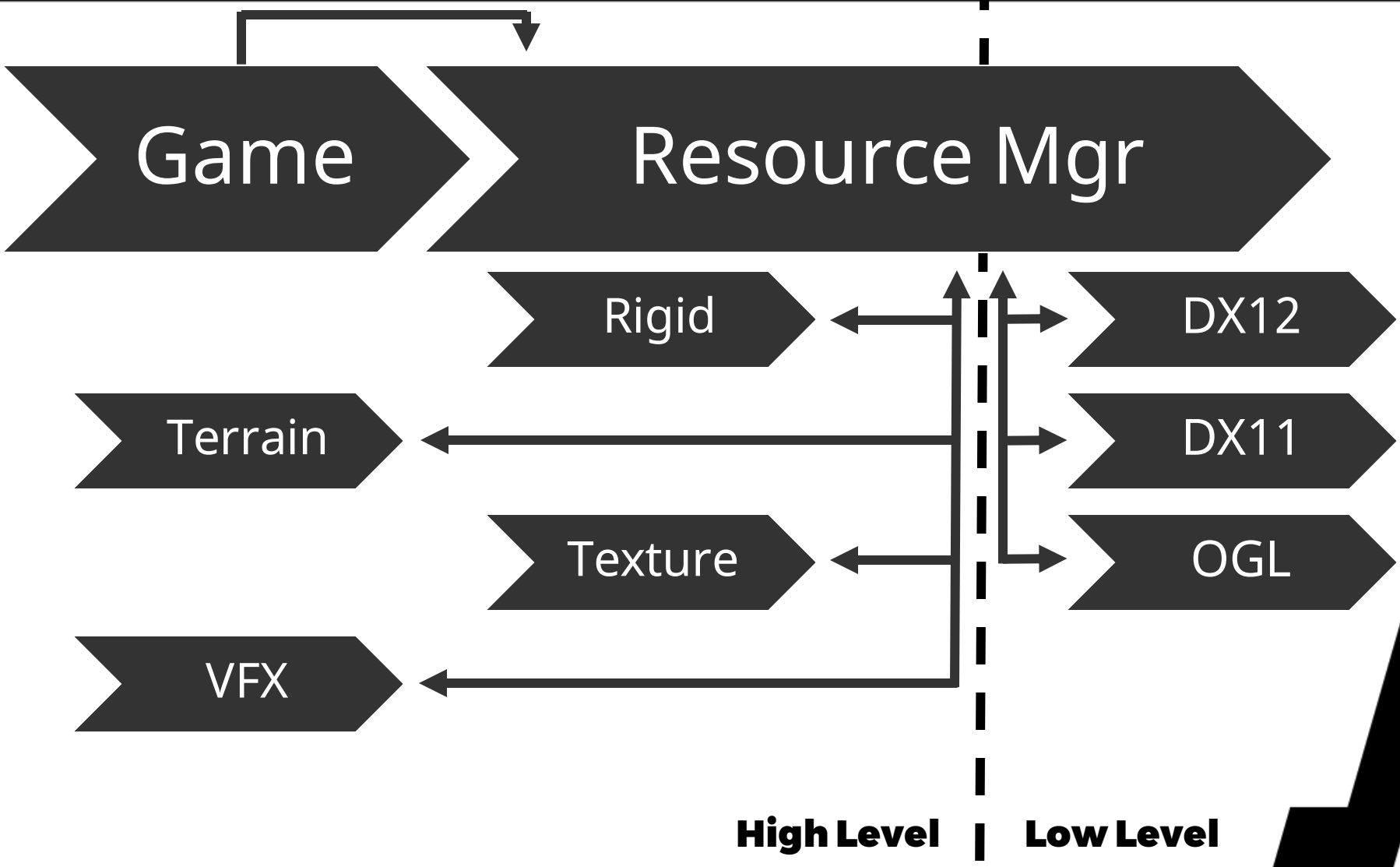
OLD PIPELINE



OLD PIPELINE



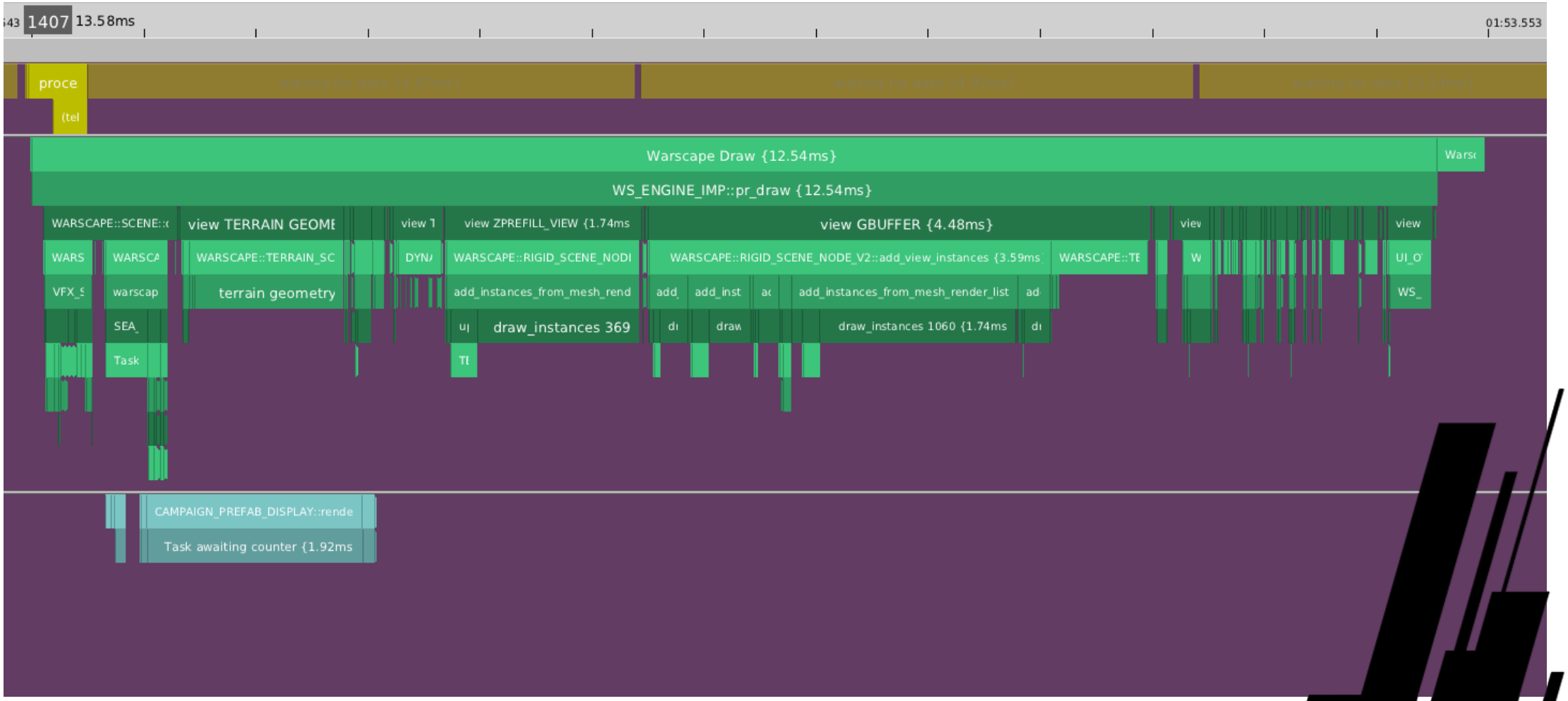
NEW PIPELINE



AGENDA

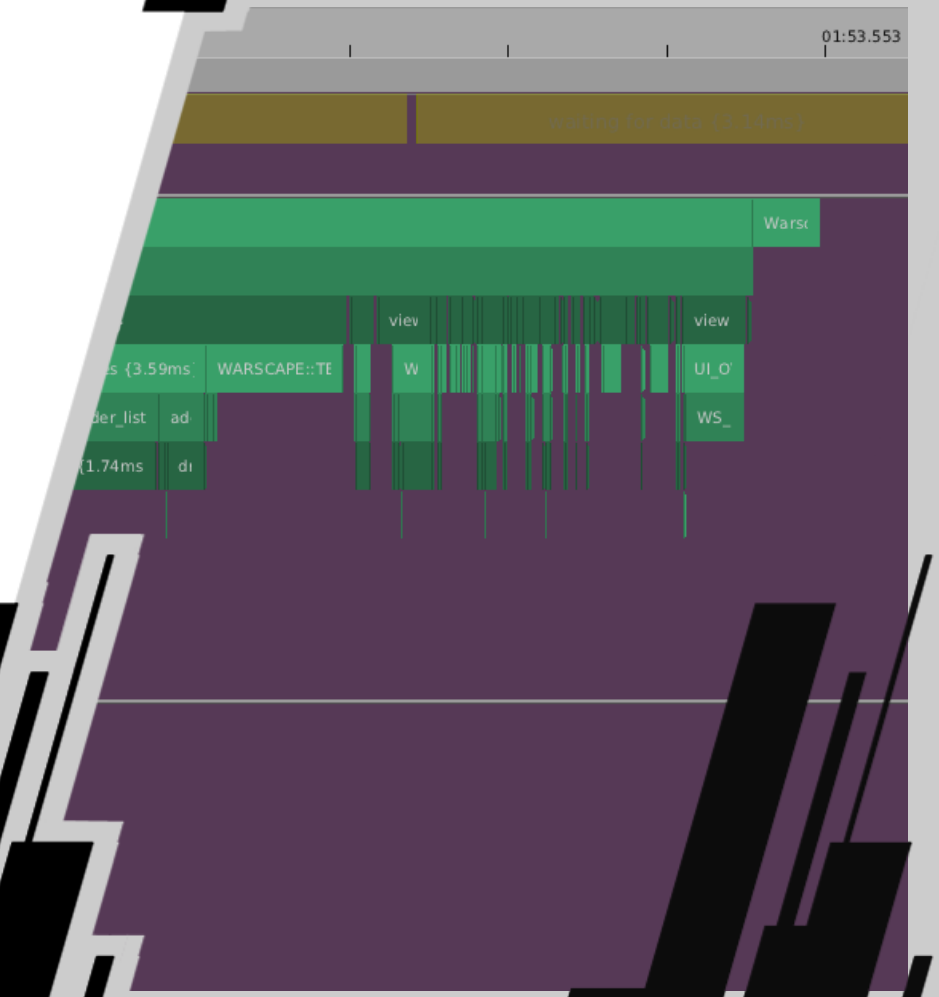
- The abstraction we had
- Problems we faced
- A better way™
- Profit
- Results

RESULTS - TW:WARHAMMER

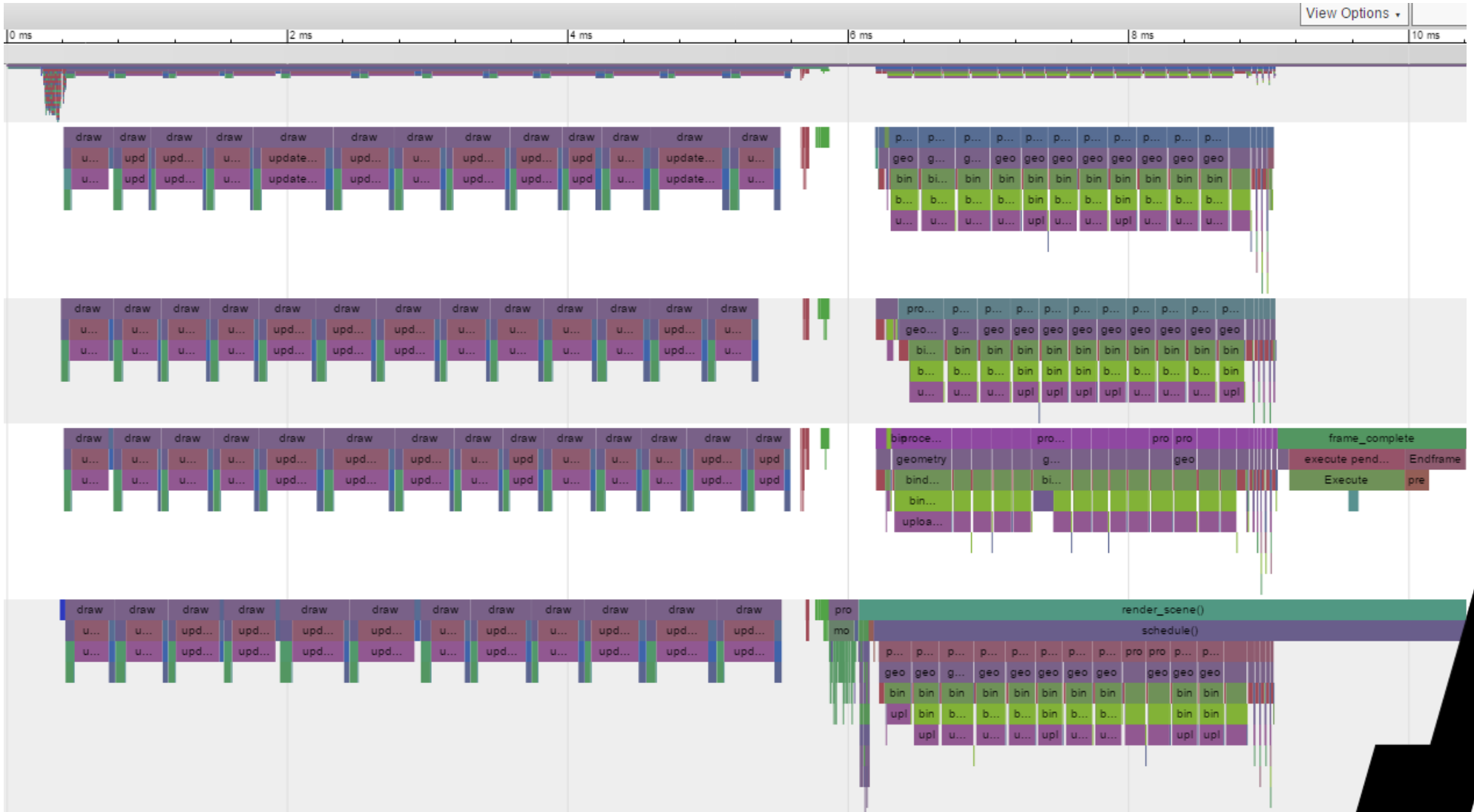


TW:WARHAMMER

- Average of 25 000 unique instances
- Single-threaded workflow with certain jobs multithreaded
- 13.5 ms on i7-4790K @ 4GHz

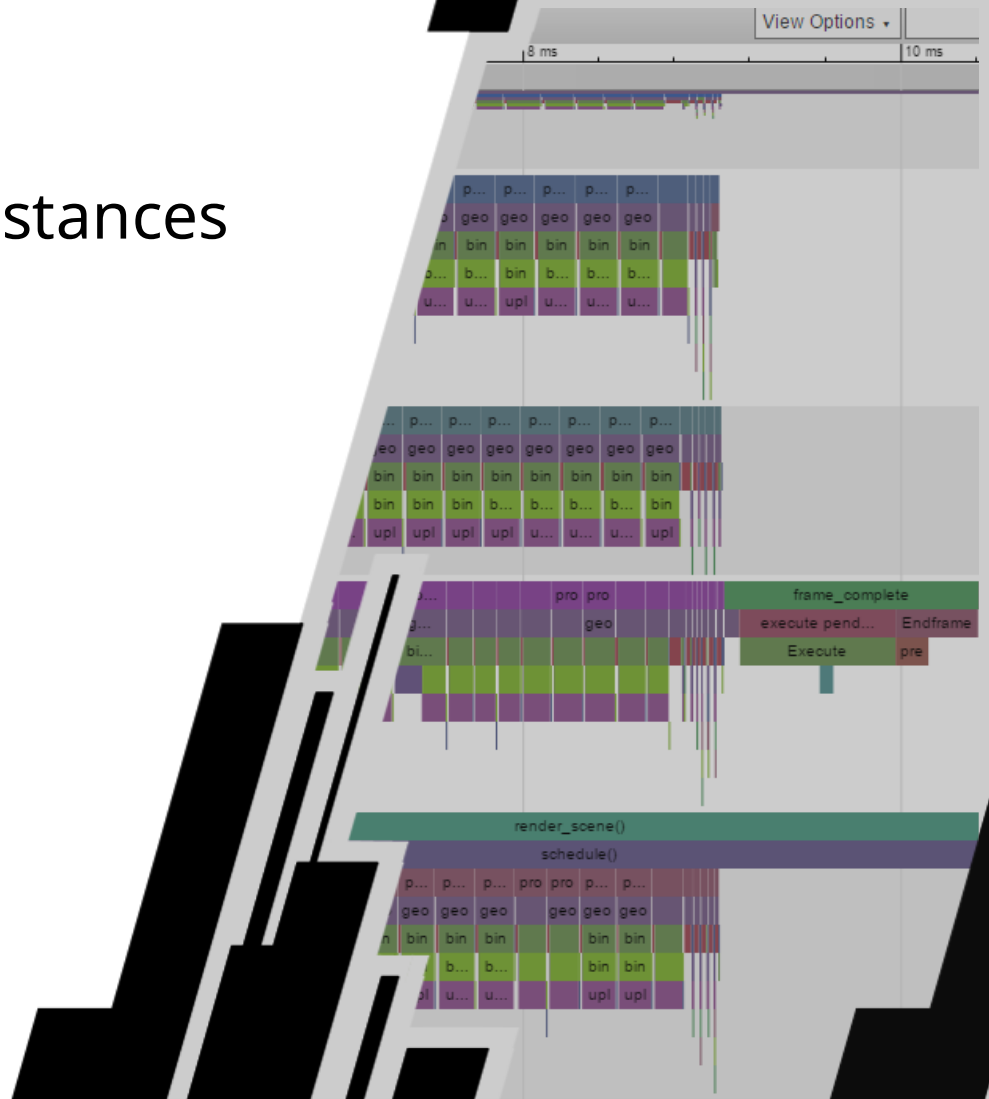


RESULTS - WARSCAPE NEXT



WARSCAPE NEXT

- Stress test of 125 000 unique instances
- Almost 100% core usage
- 9 ms on i7-4790K @ 4GHz
- Last ~1.5ms is waiting for GPU



PROFIT

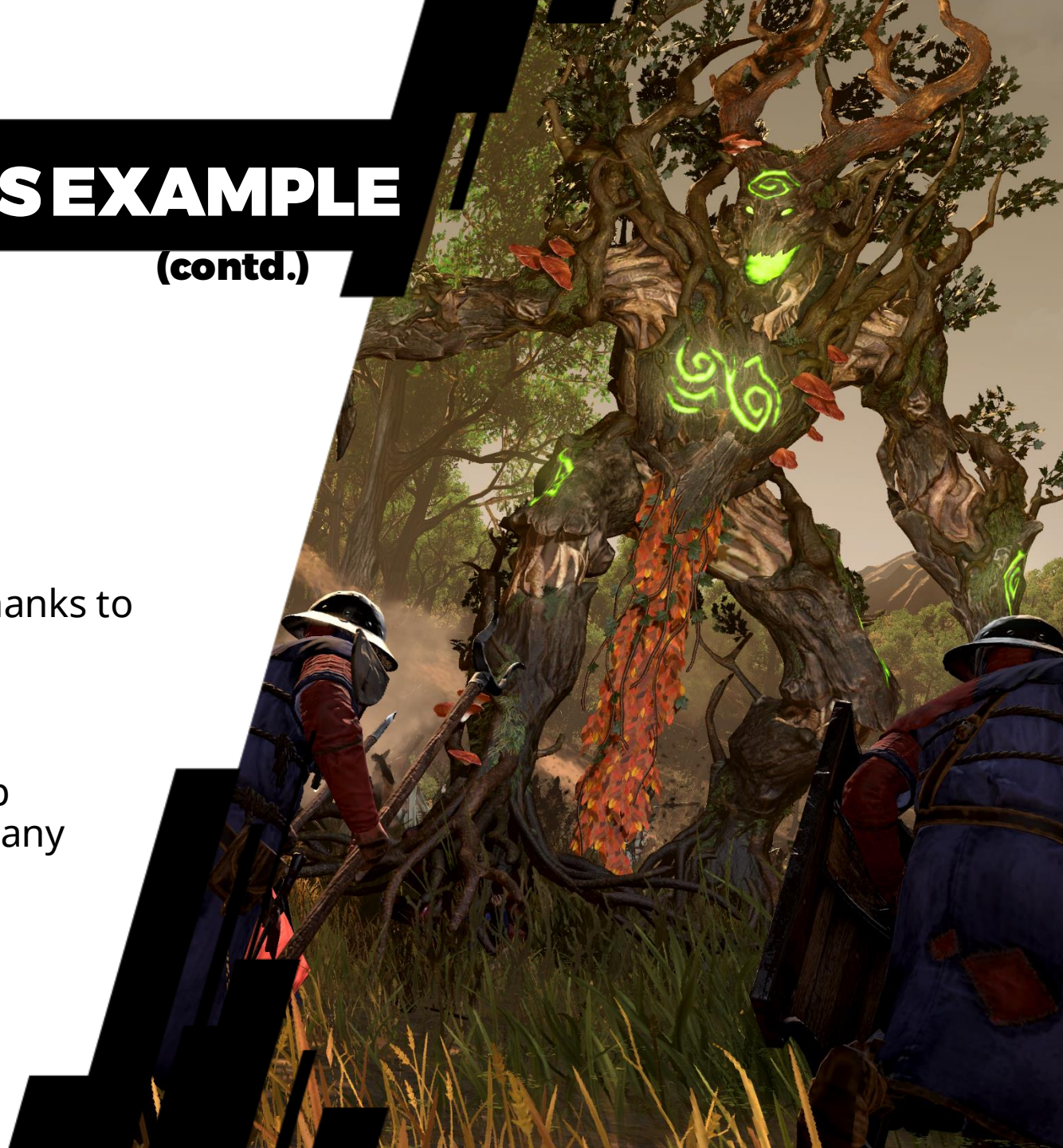
- High level code completely separated from low-level, no assumptions made
- Low level code has full control over decisions

LOW LEVEL DECISIONS EXAMPLE

- Instance data
  - **Legacy**
    - Each work item is a draw call
  - **DX11**
    - Meshes are grouped by material
    - Transform and floats are instance data
    - Textures are not instance data
    - One draw call per material group

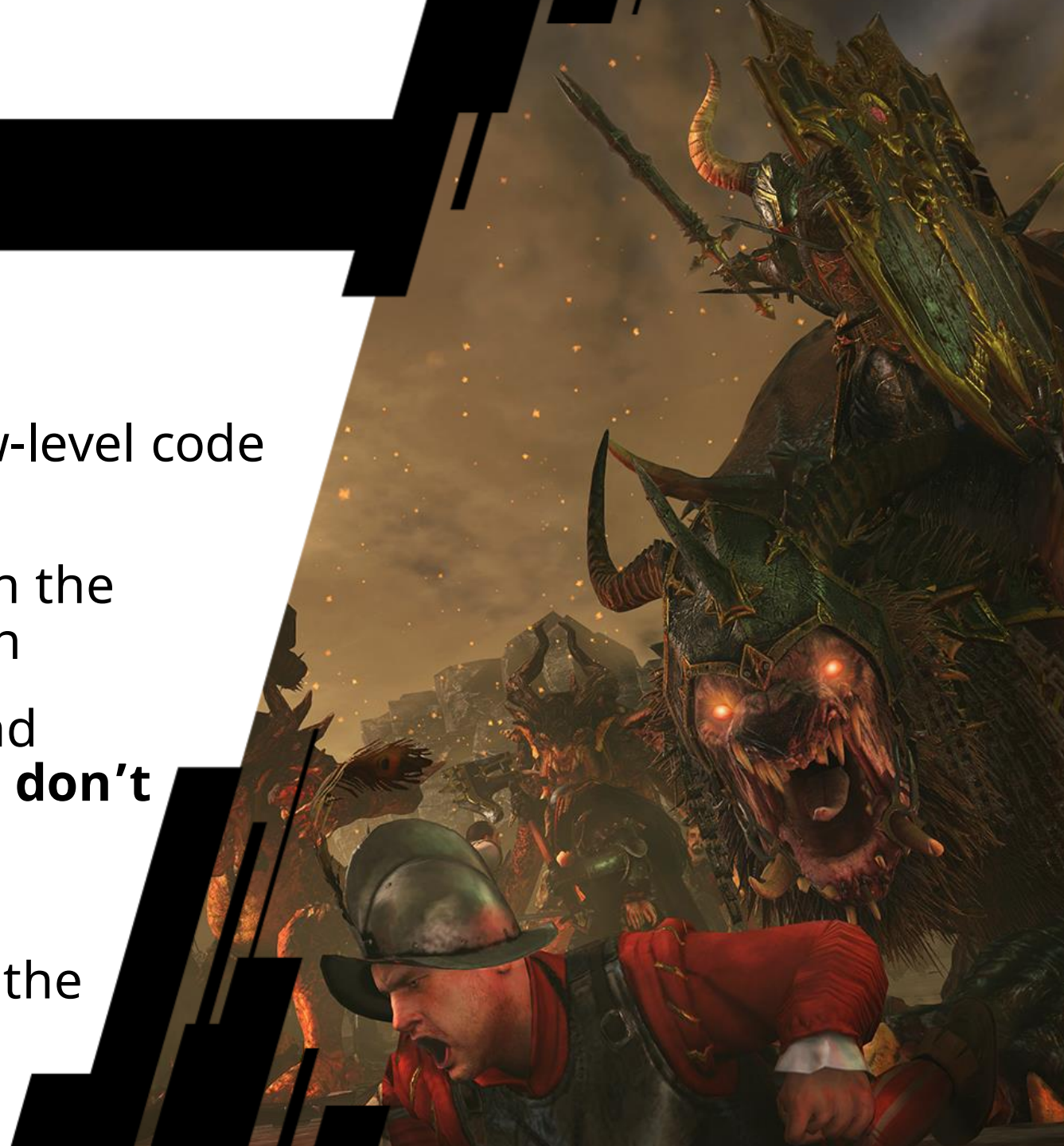
LOW LEVEL DECISIONS EXAMPLE (contd.)

- Instance data
  - **DX12**
    - Textures become instance data thanks to bindless
    - One draw call per shader
    - Per-triangle culling / triangle soup processing can be added without any changes to high level code



CONCLUSION

- Give as much context to the low-level code as possible
- Don’t make any assumptions on the high-level, keep all options open
- Don’t be afraid to generalize and abstract things **as longs as you don’t compromise performance**!
(that’s the tricky part)
- Thin wrappers over API are not the best option anymore




THANK YOU
CREATIVE ASSEMBLY
www.creative-assembly.com