

“art = space”

work in progress

Optimieren geht nicht ohne Messungen.

- Compressing von Texturen und Audio
- Effiziente File-Formate: WebP, JPEG XR OGG,AAC
 - **Texture Atlas: mehrere Texturen in einem atlas → weniger draw calls*
 - **Procedural Generation: Levels, Terrains, Texturen mit algorithmen erstellen um speicherplatz zu sparen*
 - **Texture Arrays?*
 - **Streaming assets: assets nur laden wenn sie gebraucht werden, nicht alles auf einmal, vermeidet memory verbrauch Frustum Culling, Occlusion Culling Shader: vereinfachen*
 - Auflösung und LOD: dynamisch an spieler hardware anpassen*

Vertex shading/coloring

Licht?

“Profiling”-tools: bottlenecks finden

Code Minimieren: (am Ende) wiedeerholung reduzieren,

ungebrauchtes kürzen, “tree Shaking” Effiziente Daten Strukturen, arrays statt listen, quad-trees, octrees für collision detection und culling Testen: Auf Hardware testen und probleme identifizieren Komprimieren während des Packing (LZMA, zlib)

Optimierung ist ein Prozess → sollte während der Produktion immer wieder passieren

Performance-Daten und Spielerfeedback müssen gesammelt werden um an effektiven stellen zu Optimieren

From:

<https://www.gamesforfuture.de/wiki/> - **games for future**



Permanent link:

<https://www.gamesforfuture.de/wiki/doku.php?id=optimization&rev=1714744799>

Last update: **2024/05/03 15:59**