

Trainz und die Performance

Bei einer Software wie Trainz, die von gewissenhafter Erweiterung lebt, ist es sehr schwierig möglich, die Software als solche für Probleme verantwortlich zu machen. Das bedeutet nicht, dass Trainz fehlerfrei ist. Etwa die GUI verursacht bei mir mit jedem "Update" der angezeigten Geschwindigkeit einen kurzen Stopp, was aufhört, wenn ich die GUI mittels F5 ausschalte.

Es kommt ganz oft auf eine gewissenhafte Auswahl der Objekte an. Wir mit T:ANE und TRS19 haben schon Glück, dass wir mittels 64Bit im Vergleich zu Früher mehr Ressourcen vom PC zur Verfügung gestellt bekommen, was zu mind. ein wenig das ganze kompensieren kann. Schaut man sich dazu mal im Nachbarforum zum Thema Train Simulator um, erkennt man, dass gerade diese Nutzer vom TS durch die relativ schlechte Performance des Simulators umso mehr darauf bedacht sind, effiziente und schonende Inhalte in ihren TS einzubauen. Das geht soweit, dass man sogar Autoren von zum Download angebotenen Inhalten auch mitteilt, wenn Verbesserungen möglich sind. Das haben wir mit Trainz leider noch nie richtig begriffen. Auf der [DLS](#) finden sich Objekte wie Parkbänke und Straßenlaternen mit einer Dateigröße von bis zu 100MB. Durch den einfachen Zugang dazu landen solche Objekte schnell auf der Platte, im Editor sieht man den Objekten dann ihre Dateigröße gar nicht an und verbaut es im Eifer des Gefechts auch noch auf der eigenen Strecke. Das aber nur mal als Beispiel, es gibt auch andere Möglichkeiten zur Verbesserung. Ich baue schon ewig Objekte für eine Vielzahl an Simulatoren (habe sogar schon Inhalte für Transport Fever in den Steam-Workshop gestellt).

Wenn ich ein Objekt baue, dann stelle ich mir folgende Fragen:

- Sieht man dem Objekt an, wie es sich auf die Performance auswirkt?

Beantwortet man diese Frage für sich mit nein, ist da etwas schief gelaufen!

- Inwieweit ergibt es Sinn, gerade in Bezug auf einen Eisenbahnsimulator, das Objekt so zu bauen wie ich es gemacht habe?

Jeder mag hochdetaillierte Objekte, es sieht gut aus, wenn an einem Haus Fenster und Türen ausgearbeitet sind. Aber man sollte sich trotzdem fragen,

ob es in einem Eisenbahnsimulator Sinn ergibt ein Wohnhaus so hochdetailliert zu erstellen. Man könnte es einfacher halten und dafür nützliche Funktionen, wie ein vom Wetter abhängiges Aussehen einzubauen (Ein Dach das mit Schnee bedeckt ist, zB).

- Habe ich gängige Techniken angewandt, um zu mind. ernsthaft zu versuchen, meine Arbeit zu optimieren?

In der heutigen Welt der 3D Grafik gibt es unzählige Techniken, man kann so gut wie nie versuchen die richtige Balance zwischen guten Texturen und detailliertem Mesh zu finden. Es gibt Optimierungsmöglichkeiten für beide Seiten: [LOD](#)-Stufen ermöglichen das Umschalten zwischen Meshes unterschiedlicher Detailstufe. Bibliotheken erlauben mehreren Meshes die gleiche Texturen zu nutzen, was natürlich nur dann Sinn ergibt, wenn es eine Fülle an Objekten gibt, die sich Texturen teilen (zB Vegetation).

'Scripts sind das Wundermittel'!

Nicht jedes klitzekleine Objekte muss einen Scriptthread haben, der sekundlich oder gar in der Framezeit (wobei Trainz da nen Riegel vorschiebt) irgendwelche Werte überprüft. Führerstände müssen die Update-Loop (die mit jedem Frame geschossen wird!) auslassen, wenn Sie nicht vom Spieler gesteuert werden.

Bibliotheken können nicht nur als Scriptsammlung sondern auch als eigenes Script im Spiel Threads zur Verfügung stellen und mittels PostMessage(World, ...) Statusupdates an die Spielwelt senden -> Ein Thread für sämtliche Objekte, statt 100 Objekte mit 100 Threads. Gerade wenn jemand mit Scripting anfängt erscheint oftmals als einfachster Weg ein Thread, der zur Überprüfung läuft, statt auf ein auf Ereignisse basiertes System zu setzen. Das ist logisch, weil ein Ereignisbasiertes System etwas schwieriger zu verstehen ist.

Also auch hier: Muss das Script so sein, wie ich es gemacht habe? Sollte ich vielleicht jemand anderen nochmal drüber schauen lassen für eine 2. Meinung?

Sind meine Scripts auf dem aktuellen Stand?

Die API ändert sich ständig. Kompatibilitäten brauchen Performance und tragen null zur Verbesserung bei. Gerade in TRS19 haben wir nichts vom neuen Streckenformat und nichts von der Streaming-Funktion, wenn die Software auf alte Objekte und Scripts Rücksicht nehmen muss. Zu allem Übel scheint der TRS19 komplett zu versagen, wenn man die Funktion zum Streamen einschaltet, weil nichts mehr geht. Und dann wird der Fehler in der Software gesucht. Aber das ist nicht wahr, denn: Es gibt unter meinen von N3V Games eingebauten und standardmäßig für jede Strecke geladenen Objekten nur eine Ausnahme, wo ein Script einen sog. Legacy-Call (also einfach ausgedrückt einen veralteten Funktionsaufruf, der noch zur Kompatibilität nutzbar ist) ausführt. Und das ist ein veraltetes Senden einer Nachricht.

Zum Abschluss;

Unter TS12 waren die sog. Performance Statistiken im Editor und Fahrermodus noch ein Begriff. Hier finden sich auch in TANE und TRS19 Informationen, die wertvoll sein **können**.

- Worst Script Library

Taucht da ein Script auf, dass zu einem Objekt gehört, welchem man relativ wenig Funktion zurechnet, ist hier etwas komisch! Die Zahl dahinter in Klammern gibt die Anzahl der Calls an, die das Script bisher geleistet hat. Achtung: Ein Führerstand bekommt mit jedem Frame zu mind. einen Call auf seine Update-Funktion.

- Worst Index Count & Worst Buffer Count

Das sind Werte die sich auf die Kombination der sog. Index- und Vertex-Buffer beziehen. Je kleiner das Objekt bei Worst Index Count (und je größer die Zahl dahinter), desto eher wurde verschwenderisch und ineffizient gebaut. Nicht jedes Polygon muss drei eigene Vertices haben, sondern kann sich diese auch mit anderen Polygonen teilen. Hierzu stellen die Modellierungsprogramme Funktionen zum Verschweißen von Vertices zur Verfügung.

Worst buffer count zählt die Anzahl der Vertex-Buffer, je größer die Zahl dahinter, desto mehr einzelne Meshes hat das Objekt. Das ist bei zusammengesetzten Objekten, wie Führerständen und solchen des neuen Gleistyps technisch nicht anders lösbar. Auch hier gilt: Je unbedeutender das Objekt, desto

schwerwiegender ist eine hohe Angabe an dieser Stelle.

Zitat

"Hier finden sich Informationen, die wertvoll sein **können**"

Man sieht nun, die Angaben bedeuten nicht zwingend etwas schlimmes. Es muss geschaut werden, wie viele Objekte man auf der Map so verbaut hat und dann bewerten, wie der Einfluss der Objekte, die angegeben werden, eigentlich im Vergleich zum Rest angenommen werden sollten. Beispiel: Ich fahre mit einem voll ausgestatteten und ausmodelliertem Fahrzeug mit einer Fülle an Funktionen auf meiner Strecke. Dann würde ich erwarten, dass der Führerstand eventuell in den Statistiken auftaucht. Taucht aber stattdessen Haus XY 1000m Weg von der Strecke dort auf, sollte ich mir jenes Haus nochmal anschauen und seine Position auf meiner Strecke überdenken.

Also immer schön vergleichen, denn Objekte rauswerfen weil sie dort auftauchen ist dann doch zu brachial 😊 Es geht hier nämlich um die Bewertung, nicht um das Herausfiltern von schönen Objekten.