Server-Side Renderer

Ausarbeitung zur Lösung des Blocks Spiels mittels Server-seitigem Rendering

Benedikt Lüken-Winkels - Matrikelnumber
Aaron Winziers - 1176638

Lehrstuhl für Systemsoftware und Verteilte Systeme
Universität Trier
01.04.2019

1 Einleitung

Im Rahmen der Vorlesung "Spieleprogrammierung" im Sommersemester 2019 sollten verschiedene Gebiete der Entwicklung von Computerspielen näher untersucht werden. Hierfür wurden verschiedene Gruppen gebildet, welche jeweils ein Teilgebiet oder ein Konzept untersuchen und in dem Spiel *Blocks* implementieren werden. *Blocks* war in diesem Fall ein simples "world building" Spiel in dem man sich durch eine Welt mit einem Avatar bewegen kann und in dieser Welt Blöcke setzen und löschen kann.

Diese Ausarbeitung behandelt die Implementierung eines Server-Side Renderers in der Spile-Engine Unity und erläutert die Entwicklung derselben.

2 Server-Side Renderer

Die Server-Side Renderer Architektur erklärt sich in ihrem Namen recht gut. Sie besteht aus einem Server der das Rendering eines Spiels übernimmt. Die Bilder und andere von dem Server erzeugte Daten(z.B. Audio) werden dann an einem Client geschickt und dort dem Nutzer angezeigt. Der Client ist für die Annahme und das Versenden von Nutzereingaben an den Server verantwortlich. Diese Architektur hat den Vorteil, dass der Rechenaufwand und somit einiges an Strom- un Ressourcenverbrauch von dem Client an den Server ausgelagert wird. In unserer Implementierung wurde Sie wie folgt umgesetzt:

2.1 Server

Der Server ist ein auf Unity basiertes Programm welches das *Blocks* Spiel mit seiner Initialisierung startet. Mit dem Verbinden eines Nutzers kann das Spiel sofort gespielt werden.

In der Update() Funktion des Servers wird nach jedem gerendertem Frame das erzeugte Bild mittels der EncodeToPNG() Function encodiert und über eine TCP Verbindung gesendet.

TCP wurde hier verwendet um zu versichern, dass gegeben ist, dass das vollständige Bild bei dem Client ankommt, da ein unvollständiges Bild zwar prinzipiell zu bearbeiten wäre aber nur mit Umständen die zeitlich nicht im Rahmen dieses Projektes umsetzbar waren.

Nutzereingaben die vom Client empfangen wurden in eine Queue eingefügt die ebenfalls mit jedem neuen Frame abgearbeitet wird.

2.2 Client

Der Client basiert ebenfalls auf Unity und empfängt und zeigt das Bilder Stream des Servers in dem Schaufenster des Nutzers an.

Nutzereingaben werden aufgenommen und über eine UDP verbindung an den Server gesendet. Hierfür war UDP ausreichend, da die Weise in der Nutzereingaben vom Server verarbeitet werden resistent war gegen die Auswirkungen von verlorenen Paketen.

3 Probleme undwerden Aussicht

Es gibt einige Möglichkeiten die Implementierung auf verschiedene Weisen zu verbessern.

3.1 Minimaler Client

Wie oben erwähnt ist der Client in Unity implementiert und verbraucht damit, vor allem bei dem Starten des Programms, mehr Ressourcen als notwendig wären. Da der Client nur wenig Funktionalität bieten muss, und somit nicht alle Funktionalitäten benötigt die von Unity angeboten werden, wäre es möglich und erstrebenswert ein noch weiter minimierten Client zu entwickeln.

3.2 Framerate, Auflösung, EncodeToPNG und das Netzwerkalgorithmus

Aktuell besteht die notwendigkeit die Auflösung des Spieles zu verringern um eine nützliches und vor allem spielbares Framerate zu erreichen. Hierbei gibt es zwei große Bottlenecks die großes Verbesserungspotenzial haben.

Die EncodeToPNG() Funktion ist eins dieser Bottlenecks für das Programm, da das Encodieren sehr langsam erfolgt. Da die Funktion in der Update() Funktion ausgeführt wird, muss sie vollständig ausgeführt werden vor der Nächste Frame gerendert werden kann. Um diesen Effekt zu minimieren wurde die Auflösung verringert und den Rechenaufwand zu reduzieren. Erstrebenswert wäre es eine schnellere alternative zu finden oder zu implementieren die eine größere Auflösung erlaubt mit weniger Zeitkosten.

Das zweite große Bottleneck ist die Übertragung der Bilder über eine TCP Verbindung. Ein unvollständiges Bild lässt sich nicht von Unity anzeigen also wurde, wie oben erwähnt, eine Lösung benötigt bei der die Vollständigkeit gegeben ist. Eine bessere Lösung wäre, zum Beispiel eine in der ein Best Effort Algorithmus verwendet wird in Kombination mit einem Interpolationsalgorithmus welches Lücken in dem Bild vervollständigen würde vor es angezeigt wird. Hierbei müsste jedoch beachtet werden wie Rechenintensiv dieser Algorithmus ist um zu sehen ob die erhöhte notwendige Menge an Ressourcen sich rechtfertigen lässt.

4 Anhang

Der gesamte Programmcode ist auf GitHub verfügbar.

[1] GitHub