HOCHSCHULE **TRIER**
Trier University of Applied Sciences
Informatik - Computer Science

Entwurf einer Command Queue für Echtzeit-Strategiespiele

Creating a command queue for real time strategy games

Daniel Biskup, 952976

Hausarbeit zur Vorlesung Künstliche Intelligenz für Spiele

Betreuer: Prof. Dr. Christof Rezk-Salama

Trier, 7.9.2014

Inhaltsverzeichnis

1

Einleitung

1.1 Zielsetzung

2

Konzeption der Geometrieerzeugung

In diesem Kapitel werden die Rechnungen und Definitionen hergeleitet, die benötigt werden, um rekursiv die in der Einleitung beschriebene Geometrie zu erzeugen.

2.1 Attribute der Eckpunkte

In der Computergrafik setzen sich Objekte üblicherweise aus Dreiecken zusammen. Jedes Dreieck ist über drei Eckpunkte eindeutig definiert. Häufig ist es wichtig zwischen der Vorder- und Rückseite eines Dreiecks unterscheiden zu können, zum Beispiel um Face Culling durchzuführen. Für ein auf eine Ebene projiziertes Dreieck im $\mathbb{R}^3$ kann diese Unterscheidung kann anhand der Drehrichtung in der die Eckpunkte um den Mittelpunkt des Dreiecks angeordnet sind getroffen werden. Sind die Punkte in mathematisch positiver Drehrichtung – gegen den Uhrzeigersinn – um den Mittelpunkt des Dreiecks angeordnet handelt es sich um ein Dreieck mit positiver Wicklungsrichtung (englisch. winding order) andernfalls um eines mit negativer Wicklungsrichtung. Je nach Konvention wird entweder die Seite eines Dreiecks die in negativer oder in positiver Wicklungsrichtung erscheint als Vorderseite interpretiert. In der vorliegenden Arbeit verwendete Konvention entspricht der Standarteinstellung `glFrontFace( GL_CCW );` von OpenGL, bei der die Seite mit positiver Wicklungsrichtung – gegen den Uhrzeigersinn – als Vorderseite interpretiert wird.

Daher muss bei der Generierung neuer Geometrie darauf geachtet werden, dass die Punkte der generierten Dreiecke in der richten Reihenfolge angegeben werde.

Beim Erzeugen der in der Einleitung beschriebenen Geometrie ist es wichtig, dass für jedes Dreiecken (p_0, p_1, p_2) auch die Höhe l des Pyramidenstumpfes gegeben ist, der möglicherweise aus diesem generiert werden soll. Im Folgenden wird ein Dreieck (p_0, p_1, p_2) mit Höhe l repräsentiert durch ein Tupel $p = ((p_0, p_1, p_2), l)$ mit $l \in \mathbb{N}$ wobei $\mathbb{N} := \{0, 1, 2, 3, \ldots\}$.

2.2 Konstruktion der Geometrie

Es sei gegeben, dass die Funktion zum erzeugen neuer Geometrie als Eingabe neben einem Dreieck $p = ((p_0, p_1, p_2), l)$ zusätzlich die drei über alle Iterationen hinweg

konstanten Werte `scaleTriangle`, `pyramidFactor` und `scaleLength` erwartet. Gilt $l = 0$, so soll das Dreieck unverändert zurückgegeben werden. Gilt $l > 0$, so soll die in (KABB1) abgebildete Geometrie zurückgegeben werden. Im Folgenden werden zunächst die Eckpunkte und anschließend aus diesen die Dreiecke der zu erzeugenden Geometrie konstruiert.

2.2.1 Konstruktion der Eckpunkte

Ist ein Dreieck $p = ((p_0, p_1, p_2), l)$ mit $l > 0$ gegeben, so werden zur Erzeugung der neuen Geometrie die Punkte $p_0, p_1, p_2, q_1, q_2, q_3$ und t benötigt (KABB1). Da nach gewählter Konvention gilt, dass die Vorderseite eines Dreiecks eine positive Wicklungsrichtung hat, lässt sich die Normale eines Dreiecks d über das Kreuzprodukt wie folgt bestimmen:

$$n = \frac{a \times b}{|a \times b|} \text{ mit } a = p_1 - p_0 \text{ und } b = p_2 - p_0$$

Der Schwerpunkt c_p des Dreiecks liegt bei $c = \frac{p_0+p_1+p_2}{3}$. Das Dreieck $q = (q_0, q_1, q_2)$ soll kleiner sein, als das Dreieck p. Die Eckpunkte des Dreiecks d , dessen Eckpunkte jeweils um den konstanten Faktor $0 <$ `scaleTriangle` ≤ 1 weiter von c_p entfernt sind als die von p, lassen sich bestimmen mit $d_i = (p_i - c_p) \cdot$ `scaleTriangle` für $i \in \{0, 1, 2\}$. (Siehe KABB2) Der Vektor vom Schwerpunkt c_p zu jenem des Dreiecks q ist $h = n \cdot l$.

Damit ergeben sich die Eckpunkte von q zu:

$$q_i = c_p + d_i + h \text{ für } i \in \{0, 1, 2\}$$

Die Höhe $h_{pyramid}$ der Pyramide (q_0, q_1, q_2, t) wird in diesem Projekt, ohne, dass es dazu besondere mathematische Überlegungen gäbe, festgelegt als $h_{pyramid} = |a| *$ `pyramidFactor`, wobei es sich bei `pyramidFactor` um eine Konstante handelt. Die Spitze t der Pyramide liegt damit bei $t = c_p + h + h_{pyramid} * n$.

2.2.2 Konstruktion der Geometrie aus den Eckpunkten

Wenn die Punkte $p_0, p_1, p_2, q_1, q_2, q_3$ und t bekannt sind, so kann aus diesen sowohl die Geometrie der Mantelfläche, als auch die der Pyramide erzeugt werden.

Die Dreiecke der Mantelfläche

Die Dreiecke welche die Mantelfläche des Pyramidenstumpfes bilden sollen einen Längenwert von 0 haben, damit sie bei der nächsten Iteration nicht durch weitere Geometrie ersetzt werden. Zudem sind die Eckpunkte der Dreiecke in negativer Wicklungsrichtung an zu geben, damit es auch im nächsten Schritt möglich ist die Normalen der Dreiecke zu berechnen und damit OpenGL Backface Culling durchführen kann. Wie Abbildung KAAB3 entnommen werden kann, ergibt sich die Menge der Dreiecke der Mantelfläche zu:

$$M := \{(p_0, q_1, q_0), (p_0, p_1, q_1), (p_1, q_2, q_1), (p_1, p_2, q_2), (p_2, q_0, q_2), (p_2, p_0, q_0)\}$$

Oder kürzer:

$$M := \{x \in \{(p_i, q_j, q_i), (p_i, p_j, q_j)\} | i \in \{0, 1, 2\} \wedge j = (i + 1) \bmod 3\}$$

Die Menge M_L aller Dreiecke mit Längenangabe ist damit

$$M_L = \{(d, l) | d \in M \wedge l = 0\}$$

Die Dreiecke (p_0, p_1, p_2) und q_0, q_1, q_2 sind nicht als Teil von M_L definiert worden, da sie beim betrachten der fertigen Geometrie auf dem Bildschirm ohnehin verdeckt sein würden. Das die Geometrie des fertigen Baumes auf der Unterseite des Stammes ein dreieckiges Loch haben wird, ist unproblematisch, da Bäume für gewöhnlich in der Erde stecken.

Die Dreiecke der Pyramide

Die Dreiecke welche die Pyramide bilden sollen in der nächsten Iterationen um einen Bruchteil der Länge l extrudiert werden. Daher wird als Länge für diese Dreiecke $l_{next} = l * \texttt{scaleLength}$ gewählt mit $0 \leq \texttt{scaleLength} \leq 1$. Die Menge P der Dreiecke der Pyramide lässt sich beschreiben durch $P := \{(q_0, q_1, t), (q_0, q_1, t), (q_0, q_1, t)\}$ (siehe KABB4). Die Menge P_L der Dreiecke der Pyramide mit Längenangabe ist damit $P_L = \{(d, l_{next}) | d \in P \wedge l_{next} = l * \texttt{scaleLength}\}$.

Die Menge D aller Dreiecke der in (KABB1) abgebildeten Geometrie ergibt sich damit zu $D = M_L \cup P_L$. D hat eine Mächtigkeit von neun.

2.3 Herleitung der Formel zur Berechnung der Anzahl generierter Dreiecke nach n Iterationen

Beim absetzen eines Draw Calls an OpenGL über die Methode `void glDrawArrays( GLenum mode, GLint first, GLsizei count);` muss als dritter Parameter angegeben werden, wie viele Eckpunkte zu zeichnen sind. Da die in diesem Kapitel generierte Geometrie aus einzelnen Dreiecken zusammensetzt, und jedes Dreieck sich aus drei Eckpunkten besteht, ist die Anzahl der Eckpunkte in der Geometrie dreimal so groß wie die der Dreiecke. Um die Anzahl der Eckpunkte in der Geometrie zu bestimmen ist es daher hilfreich zunächst die Anzahl der Dreiecke aus denen diese sich zusammensetzt zu bestimmen.

Nach n Iterationen besteht die Geometrie des generierten Baumes aus $4 * 3^n - 3$ Eckpunkten. Diese Formel wird im Folgenden hergeleitet.

Herleitung

Für jedes Eingabedreieck $p = ((p_0, p_1, p_2), l)$, für welches $l = 0$ gilt wird keine neue Geometrie erzeugt, die Gesamtzahl der Dreiecke bleibt unverändert. Gilt $l > 0$

so wird das Eingabedreieck durch neun andere Dreiecke ersetzt, die Gesamtzahl steigt um acht Dreiecke.

In jeder Iteration wird jedes Dreieck mit $l > 0$ ersetzt durch drei neue Dreiecke mit $l > 0$. Also enthält die Eingabemenge der Dreiecke nach n Iterationen genau 3^n Dreiecke mit $l > 0$. Da mit jeder Iteration für jedes Dreieck mit $l > 0$ genau acht neue Dreiecke hinzukommen, und nach m Iterationen 3^m Dreiecke in der Eingabemenge enthalten sind die diese Bedingung erfüllen, kommen in Iteration n genau 3^n Dreiecke hinzu für $n \geq 0$. Für die Anzahl der Dreiecke f die in Iteration n hinzukommen gilt daher $f(n) := 3^n * 8$.

Die Gesamtzahl $g(n)$ der Dreiecke nach n Iterationen ist damit:

$$
\begin{aligned}
g(n) &:= 1 + \sum_{i=0}^{n-1} f(i) \\
&= 1 + \sum_{i=0}^{n-1} 8 * 3^i \\
&= 1 + 8 * \sum_{i=0}^{n-1} 3^i \\
&= 1 + 8 * ((\sum_{i=0}^{n} 3^i) - 3^n) \\
&= 1 + 8 * (\frac{1 - 3^{n+1}}{1 - 3} - 3^n) \\
&= 1 + 8 * (\frac{1 - 3^{n+1}}{-2} - 3^n) \\
&= 1 + 8 * (\frac{1 - 3^{n+1} + 2 * 3^n}{-2}) \\
&= 1 + 8 * (\frac{1 - 3 * 3^n + 2 * 3^n}{-2}) \\
&= 1 + 8 * (\frac{1 - 3^n}{-2}) \\
&= 1 + (\frac{4 * (1 - 3^n)}{-1}) \\
&= 1 - (4 - 4 * 3^n) \\
&= 4 * 3^n - 3
\end{aligned}
$$

Da für $n = 0$ gilt $g(0) = 4 * 3^0 - 3 = 1$, gilt:

$$\forall n \in \mathbb{N} : g(n) := 4 * 3^n - 3$$

3

Implementierung

Dieses Kapitel behandelt das diesem Dokument beiliegende Programm. Zunächst wird Überblick darüber gegeben, wie das Programm zu bedienen ist. Anschließend wird auf die verwendeten Libraries, und den Aufbau des Programmes eingegangen. Zuletzt wird ausgeführt wie das Programm auf den Plattformen Windows und Linux kompiliert werden kann.

3.0.1 Bedienung des Programmes

Nach dem Start des Programmes ist in der Mitte des Fensters ein sich drehender Baum zu sehen. In der linken oberen Ecke befindet sich ein Fenster mit der mit dem Titel "TweakBar"welches im folgenden als Tweak Bar bezeichnet wird.

Die Tweak Bar enthält drei Gruppen von Einträgen, mit Namen "generation parameters", "presentation parametersünd "read-only scene information".

Während die ersten Beiden Gruppen Felder die vom Benutzer oder der Benutzerin manipuliert werden können enthalten, enthält die letzte Gruppe die nicht vom Benutzer oder der Benutzerin nicht direkt manipulierbaren Einträge "number of trianglesünd "number of vertices". Unter "number of trianglesïst die Anzahl der Dreiecke und unter "number of verticesänderen die Anzahl der Eckpunkte aufgeführt, aus denen sich der aktuell auf dem Bildschirm angezeigt Baum zusammensetzt.

In der Gruppe "presentation parameters"findet der Benutzer oder die Benutzerin den Schalter äutoRotation"mit dem sich festlegen lässt, ob der Baum automatisch entlang seiner Y-Achse gedreht werden soll, sowie das Eingabefeld äutoRotationSpeed"über welches die bei automatischer Rotation anzuwendende Rotationsgeschwindigkeit festgelegt werden kann. Über den Eintrag "rotation"kann durch klicken und ziehen der bunt eingefärbten Kugel die Orientierung des Baumes im Raum verändert werde.

In der Gruppe "generation parametersßind Einträge aufgeführt über die sich der Prozess der Geometriegenerierung des Baumes konfigurieren lässt. Den ersten Eintrag bildet ein Button mit Aufschrift "click here to generate tree". Ein Klick auf diesen löst die Ausführung des in [Kapite X] beschriebenen Geometrieerzeugungsalgorithmus aus, welchem als Parameter die Werte der übrigen Eingabefelder, die in der Gruppe "generation parametersënthalten sind, übergeben werde. Wurde die

Geometrie erfolgreich erzeugt, so wird der diese nun anstelle der zuvor angezeigten angezeigt.

Der Parameter "numberOfIterations"gibt an, wie viele Iterationen des Geometrieerzeugungsalgorithmus verwendet werden sollen.

Die übrigen Parameter sind bereits aus [Kapitel X] bekannt, ihr Bedeutung soll hier dennoch kurz geklärt werden.

Der Parameter ßcaleLength"bestimmt, wie lang ein Astsegment im Verhältnis zu seinem Vorgänger ist.

Der Parameter ßcaleTriangle"bestimmt, wie dick das obere Ende eines Astsegmentes im Verhältnis zu seinem Unteren Ende ist.

Der Parameter "pyramidFactor"bestimmt, wie

3.0.2 Verwendet Libraries

3.0.3 Aufbau des Programmes

3.0.4 Der Build Prozess