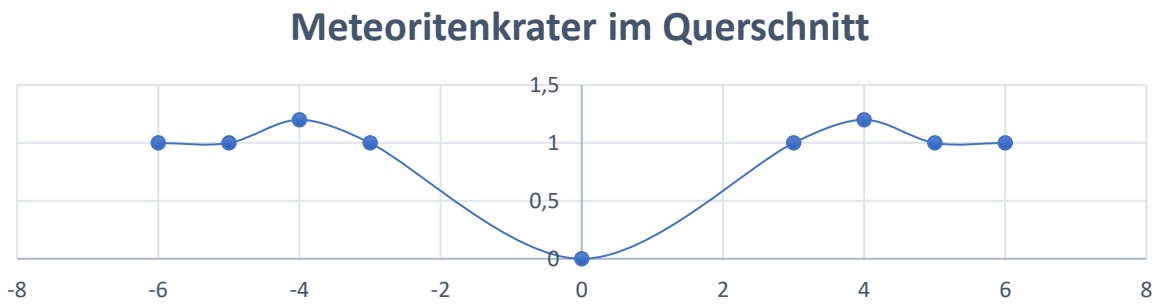
Prozedurale Generierung

Angewandte Mathematik

In der Informatik gibt es ein Verfahren, dass sich Prozedurale Generierung nennt. Das ist ein Verfahren, welches unteranderem in Videospielen verwendet wird um ohne das Zutun von Spielegrafik-Designern automatisch voll funktionsfähige Spielwelten oder Texturen zu generieren. In einem neuen Videospiel soll nun ein Meteoritenkrater generiert werden. Aber wie bringt man das einem Programm bei? Die Antwort findet sich wie so oft in der Mathematik.

Es gibt verschiedene Verfahren. Sogenannte Interpolationsverfahren können verwendet werden, um ein Polynom zu errechnen, welches bekannte Punkte miteinander verbindet. Da man hierbei allerdings keine Kontrolle über die Form des Graphen hat, ist es wesentlich besser, beispielsweise das Gauß-Verfahren zu nutzen um ein Polynom, das einem Meteoritenkrater gleicht, zu berechnen. Bei dieser Methode gibt es allerdings das Problem, dass der Grad des Polynoms sehr hoch sein kann, was die Berechnung stark erschwert. Für diesen Fall behilft man sich mit der Trassierung, oder auch bekannt unter dem Namen abschnittsweise definierte Funktionen. Hierbei werden einzelne Teilfunktionen berechnet, die am Schluss den gesamten Graphen ergeben.



Betrachten wir nun den Querschnitt eines Kraters im Bereich $x \in [0;\ 3]$. Aus diesem lassen sich folgende idealisierte Daten herauslesen:

$$f(3) = 1;\ \ f(0) = 0;\ \ f'(3) = 1;\ \ f'(0) = 0$$

Mit Hilfe der vier gegebenen Werte kann nun ein Polynom dritten Grades bestimmt werden. Dabei werden die ersten beiden in die Grundform eingesetzt und der dritte und vierte Wert in die Ableitung der Grundform. Anschließend werden die aufgestellten Gleichungen vereinfacht.

$$a \cdot 3^3 + b \cdot 3^2 + c \cdot 3 + d = 1 \qquad 27a + 9b + 3c + d = 1$$
$$a \cdot 0^3 + b \cdot 0^2 + c \cdot 0 + d = 0 \qquad d = 0$$
$$3 \cdot a \cdot 3^2 + 2 \cdot b \cdot 3 + c = 1 \qquad 27a + 6b + c = 1$$
$$3 \cdot a \cdot 0^2 + 2 \cdot b \cdot 0 + c = 0 \qquad c = 0$$

Nun kann das aufgestellte Gleichungssystem zum Beispiel mit dem Gauß-Verfahren gelöst werden. Zum Schluss ergibt sich die gewünschte Teilfunktion.

$$f(x) = \frac{1}{27}x^3 \ \ \text{für}\ x \in [0;\ 3]$$

Im folgenden Beispiel ist die y-Achse nach oben orientiert. Nachdem die Grundfunktion bzw. die Grundfunktionen berechnet wurden, müssen diese im Raum dargestellt werden. Dafür gibt es ebenfalls mehrere Möglichkeiten, beispielsweise die Verwendung von Paraboloiden. Die beste Option ist in diesem Fall jedoch die Methode der Distanzberechnung, bei der die folgenden verallgemeinerten Funktionen verwendet werden.

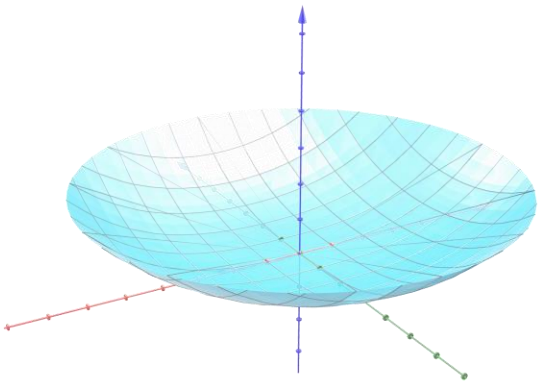
$$f\big(d_M(x;z)\big) = a_n\big(d_M(x;z)\big)^n + a_{n-1}\big(d_M(x;z)\big)^{n-1} + \cdots + a_2\big(d_M(x;z)\big)^2 + a_1\big(d_M(x;z)\big) + a_0$$

$$d_M(x;z) = \sqrt{(x - x_M)^2 + (z - z_M)^2}$$

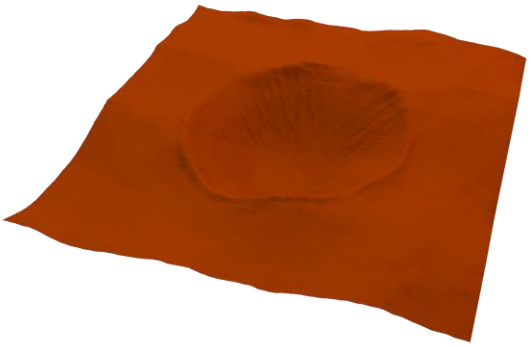
Dabei wird der Mittelpunkt des Meteoritenkraters beispielsweise auf $M(0|0|0)$ festgelegt und um den Funktionsgraphen richtig darstellen zu können, wird die Differenz der x und z-Koordinaten gebildet und mit Hilfe des Satz des Pythagoras die Entfernung zum Mittelpunkt berechnet. Anschließend wird auf den resultierenden Wert die berechnete Teilfunktion angewandt und es ergibt sich folgende Funktion.

$$f(x;z) = \frac{1}{27}\left(\sqrt{(x-0)^2 + (z-0)^2}\right)^3$$

Diese Teilfunktion sieht im Raum dann wie folgt aus:



Wenn nun alle Teilfunktionen mit Hilfe des gerade beschriebenen Verfahrens bestimmt wurden, kann der gesamte Graph dargestellt werden. Anschließend ist noch etwas programmier-Geschick gefragt, um den Krater zum Beispiel in der Unity-Engine – das ist eine Entwicklungsumgebung für Videospiele – darzustellen.



Ein Einschlagskrater eines Meteoriten könnte in einem realen Videospiel dann in etwa so aussehen:

Zusammenfassend kann man sagen, dass in der Programmierung auch oft nur simple quadratische Funktionen verwendet werden, um Ressourcen zu schonen und Berechnungszeiten zu verkürzen. Außerdem wird in Computersystemen zum Lösen von Ebenengleichungen kein Gauß Verfahren verwendet, sondern eine computerfreundliche Alternative.

Weitere Materialien sowie ein Literaturverzeichnis finden sich unter:
https://github.com/FelixDoubleu/Unity-Procedural-Generation