The Next Level
Minecraften für Profis

DIE LESEPROBE

ALLES IST ILLUSION – HINTER DEN KULISSEN VON MINECRAFT

Minecraft ist faszinierend, weil es so vielfältig ist. Die bekannte Survival-Welt mit ihren Landschaften, Tieren, Monstern, Bauwerken und sogar anderen Dimensionen hast du allerdings schon ausgiebig erforscht. Das Überleben ist längst keine Herausforderung mehr für dich, und vielleicht spielst du sogar seltener Minecraft als früher.

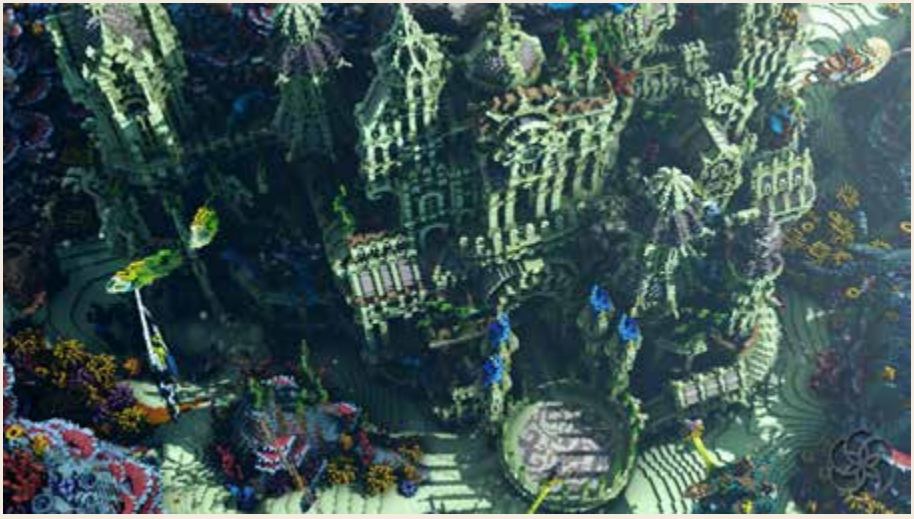
Nicht verzagen! Dieses Buch zeigt dir eine ganz neue Welt hinter den Kulissen des Spiels. Minecraft bietet eine Vielzahl von Möglichkeiten, das Spiel kreativ zu verändern. Hier erfährst du, wie du mit Texturen, Mods, Shadern und weiteren Zusätzen dein Survivalspiel völlig neu erlebst und wie du im Kreativmodus mit vielen Befehlen eigene Abenteuerwelten erschaffst.

ÜBER DIESES BUCH

Willkommen im nächsten Level! Hier lernst du spannende Dinge kennen, die Minecraft schicker und aufregender machen: Skins tauschen, Ressourcen- und Datenpakete laden, Add-ons aktivieren, Mods und Shader installieren – du erfährst, was das ist und wie du das Spiel damit veränderst. Denn darum geht es in diesem Buch: um das kreative Verändern des Spiels. Mit diesem Buch probierst du alternative **Survival-Welten** aus oder spielst im **Abenteuermodus** ganz andere Spiele mit Minecraft.

Dazu gehört natürlich auch der **Kreativmodus**, der für das Gestalten von Welten gedacht ist. Im Kreativmodus spielt Survival keine Rolle mehr. Du kannst fliegen und ohne Hunger und Schaden Paläste, Skulpturen oder ganze Städte erschaffen.

Ein Motorrad, gebaut von K_San_ auf Unlimitedworld.de



»Echoes from the Deep«, gebaut von der Gruppe EverbloomStudios

»Nova Celes City«, gebaut von der Gruppe QuanticsBuild

Das ist aber noch längst nicht alles: Im Kreativmodus lernst du viele Minecraft-Befehle kennen und zusätzliche Blöcke und Gegenstände, die es im Überlebensmodus nicht gibt. Damit öffnet sich für dich eine ganz neue Art, Minecraft zu spielen: Jetzt baust du deine eigenen Abenteuerszenarien, z. B. eine Spukvilla, ein Kreuzfahrtschiff oder Minigames. Und das Tolle ist: Mit Minecraft bist du immer mittendrin in deinen Bauwerken und kannst sie alleine oder mit Freunden erleben.

Wie war das noch mal mit dem Survival?

Der Vorgänger des »Next Level«-Buches ist das **»Survival-Buch«**, das alle Aspekte des Überlebensspiels abdeckt: Biome entdecken, Nahrung beschaffen, Bergbau betreiben, Blöcke und Gegenstände craften, Färben, Schmieden, Farmen anlegen, Tiere vermehren, Waffen herstellen und verzaubern, gegen Monster kämpfen – auch die schwierigen –, imposante Gebäude errichten, Reiten, Boot fahren, Schienensysteme anlegen, generierte Bauwerke erforschen, mit Dorfbewohnern handeln, Nether und Ende erkunden, Tränke brauen, einfache und komplexe Redstone-Schaltkreise konstruieren und auf Multiplayer-Servern spielen – das und mehr beschreibt das Survival-Buch mit vielen Tipps und Tricks. Und im Anhang gibt es eine Übersicht über sämtliche Blöcke und Gegenstände (*www.rheinwerk-verlag.de/5510*).

Java und Bedrock

Minecraft gibt es für unterschiedliche Geräte. Für den Computer bekommst du automatisch gleich zwei Versionen, die *Java Edition* und die *Bedrock Edition*, die auch »Minecraft für Windows« genannt wird.

Beide Versionen kannst du über den Minecraft-Launcher starten. Sie haben beide einen nahezu gleichen Inhalt, unterscheiden sich aber stark im Angebot von Spielzusätzen:

- Die **Bedrock Edition** hat im Spiel einen *Marketplace*, in dem vor allem kostenpflichtige Zusätze verkauft werden.
- Die **Java Edition** ist viel freier, sie hat keinen Shop im Spiel. Stattdessen kannst du eine riesige Auswahl an Spielzusätzen aus dem Internet herunterladen, die von anderen Spielern stammen und kostenlos sind. Du musst sie nur selbst suchen und installieren, wobei dir dieses Buch natürlich hilft. Die beliebten selbst programmierten Spielmodifikationen (*Mods*) sind überhaupt nur mit der Java Edition möglich. Auch bei den Minecraft-Befehlen im Spiel bietet sie mehr Möglichkeiten.

Daher geht dieses Buch vor allem auf die Java Edition ein und beschreibt in den nächsten Kapiteln, wie du damit das Spiel kreativ veränderst. Für die Bedrock Edition gibt es gegen Ende des Buches ein eigenes Kapitel, das deren Möglichkeiten zusammenfasst und beschreibt, wie du auch ohne Marketplace an kostenlose Zusätze herankommst.

Möchtest du sofort loslegen, reicht es, wenn du jeweils nur den Anfang der Kapitel liest. Möchtest du eine Methode genauer kennenlernen und damit experimentieren, liest du in dem entsprechenden Kapitel weiter bis zum Schluss.

Die Kapitel sind aufeinander aufgebaut, d. h., in späteren Kapiteln wird das Wissen aus früheren Kapiteln vorausgesetzt. Du kannst trotzdem einfach mittendrin einsteigen. Stolperst du dabei über unbekannte Begriffe, findest du die Erklärung dazu über den **Index** ganz am Ende des Buches. An manchen Stellen erinnert dich ein kleines ⓘ daran, in den Index zu schauen.

Aber noch stehen wir ganz am Anfang. Das allererste Kapitel führt dich hinter die Kulissen des Spiels und erklärt, wie Minecraft die Illusion einer belebten Welt erzeugt. Wenn du die Tricks des Spiels kennst, verstehst du besser, wie es funktioniert und wie du es verändern kannst.

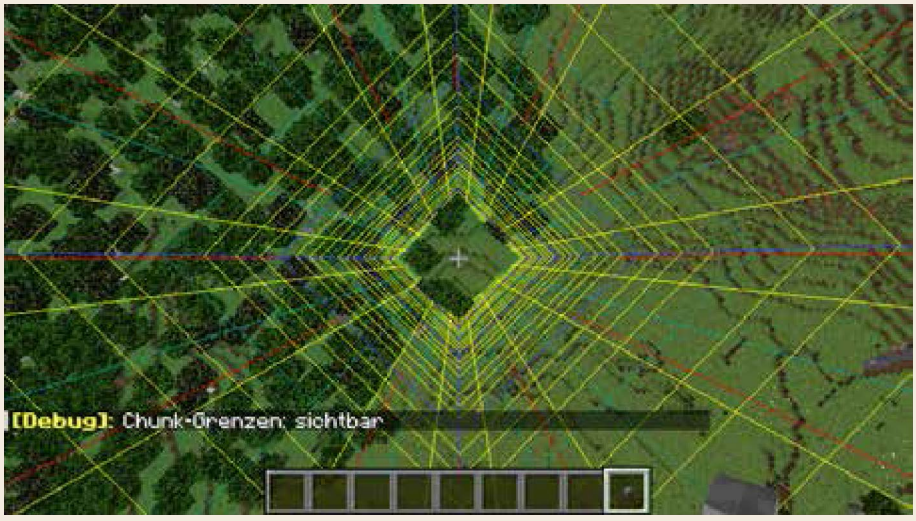
Drücke nun Esc und verlasse die Welt. Viele denken, hier ist das Spiel zu Ende. Doch jetzt wird es erst richtig interessant. Was passiert eigentlich mit der Welt, wenn du sie verlässt?

DIE WELT IST EINE ILLUSION – SIE IST NUR DA, WO DU BIST

Wenn du deine Einzelspielerwelt mit Esc verlässt, wird sie *gespeichert*. Alles friert ein: Die Zeit steht still, jeder Block bleibt in seinem augenblicklichen Zustand, und jede Kreatur verharrt in ihrer Position. Selbst Geschosse oder ein fallender Block wie Sand bleiben mitten in der Luft stehen. So kannst du die Welt stunden- oder tagelang verlassen. Wenn du sie wieder betrittst, wird der gespeicherte Zustand geladen, und alles geht weiter, als ob du nie weg gewesen wärst.

Welten können riesig sein, das Speichern würde bei solchen Datenmengen sehr lange dauern. Warum geht es trotzdem immer so schnell? Die Antwort lautet: Die Welt reicht nur bis zum Horizont. Dahinter ist nichts!

Das Spiel wird durch deinen Computer gesteuert. Damit er nicht überlastet wird, kümmert er sich nur um den Teil der Welt, den du siehst – festgelegt durch die **Sichtweite**, die du in den Minecraft-Grafikoptionen einstellst. Als Maßeinheit steht dort »Chunks«. Ein *Chunk* ist ein Ausschnitt der Welt mit einer Grundfläche von 16 × 16 Blöcken. In der Java Edition siehst du die Chunks mit F3 + G. Bei Anzeigeproblemen lädst du mit F3 + A alle Chunks neu, ohne die Welt verlassen zu müssen.



Mit F3 + G machst du die Chunk-Grenzen sichtbar.

Was F3 alles kann

F3 schaltet in der Java Edition den *Debug-Bildschirm* ein (auf Laptops oft Fn + F3, auf Macs alt + fn + F3). In der Bedrock Edition gibt es das nicht.

Programmfehler werden englisch *Bug* (»Wanze«) genannt, die Fehlerbeseitigung heißt *Debugging* (»Entwanzen«). Zur Eingrenzung von Fehlern sind zusätzliche Informationen für die Entwicklerinnen und Entwickler hilfreich. Diese liefert ihnen der Debug-Bildschirm.

Der Debug-Bildschirm. Wichtig sind vor allem die Koordinaten bei »Block«. Sie bezeichnen die Fußposition deiner Spielfigur. In der Bedrock Edition blendest du die Koordinaten über Esc ▪ »Einstellungen« ▪ »Spiel« ▪ »Weltoptionen« dauerhaft ein.

Für Survival-Spieler gibt es im Debug-Bildschirm nur wenig interessante Informationen. Die wichtigste sind natürlich die Koordinaten. Anders sieht es aus, wenn du dich mit den Themen dieses Buches beschäftigst. Da bietet F3 einige interessante Details, die in den nachfolgenden Kapiteln genauer beschrieben werden.

Dazu gehören auch zusätzliche Tastenkombinationen. Eine Liste erhältst du mit F3 + Q. Für Erklärungen zu allen Tasten schau in den Index am Ende des Buches unter »F3«.

Möchtest du das Spiel mit Esc anhalten, um die F3-Informationen genauer anzusehen, stört das Spielmenü. Drückst du F3 + Esc, hält das Spiel ohne Menü an.

Im Hauptspeicher (*RAM*) deines Computers befindet sich nur der sichtbare Ausschnitt der Welt. Ein Nebel am Horizont verbirgt, dass die Welt dort endet. Der Rest der Welt ruht auf dem Festspeicher, aber das kannst du nicht sehen, weil das Spiel einen Trick anwendet: Bewegst du dich Richtung Horizont, wird der Teil, den du verlässt, automatisch gespeichert und der Teil, auf den du

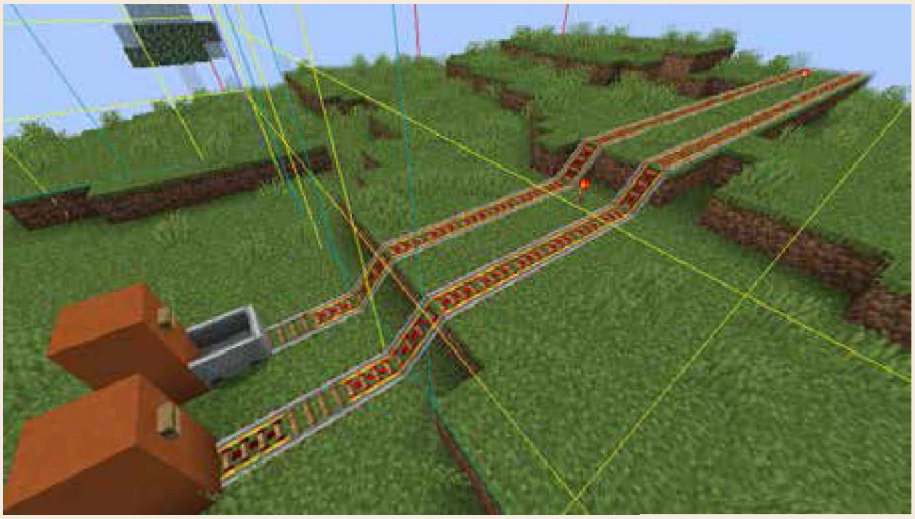
zuläufst, entweder neu generiert (wenn du ihn das erste Mal betrittst) oder wieder in den Hauptspeicher geladen (wenn du schon mal dort warst). Weil das Speichern und Laden stets außerhalb deiner Sichtweite passiert und die Datenmenge relativ klein ist, merkst du es nicht.

Hier ist die Sichtweite auf nur vier Chunks eingestellt. Im unteren Bild ist der Nebel mit der OptiFine-Mod ausgeschaltet, dadurch siehst du die Kanten der Welt. An den Ecken erkennst du, dass die Welt chunkweise geladen wird.

Mit diesem Wissen kannst du auch die Frage beantworten, wie weit eine Lore fährt: Maximal bis zum Horizont, weiter geht es nicht. Probiere das einmal aus:

1. Stelle zuerst deine Sichtweite auf den Minimalwert 2 Chunks (Bedrock 5) und betritt dann eine Einzelspielerwelt.
2. Schalte F3 + G ein, lege aktivierte Antriebsschienen mindestens drei Chunks weit (BE sechs), baue eine Kehrtwende und führe die Strecke zurück zum Start.

3. Setze dich in eine Lore und fahre einmal im Kreis. Das funktioniert problemlos, weil die Welt immer da ist, wo du bist.
4. Schicke die Lore alleine los und warte auf ihre Rückkehr. Sie wird nicht zurückkommen, weil sie am Horizont – am Ende der Welt – festhängt.
5. Gehe hinterher, und sie kommt zurück, weil jetzt der Chunk mit der Kehrtwende geladen wird.



Sitzt du nicht in der Lore, fährt sie nur bis zum Horizont.

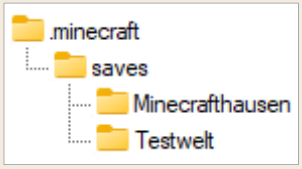
Je größer du die Sichtweite einstellst, desto größer ist die Belastung deines Hauptspeichers, denn die Fläche und damit die Anzahl der geladenen Chunks wachsen quadratisch. Die Anzahl der aktuell geladenen Chunks siehst du im Debug-Bildschirm unter »Chunks [C] W«. Dort stehen zwei durch Komma getrennte Zahlen, es ist die zweite Zahl.

Chunks siehst du auch an einer anderen Stelle im Spiel: In der Java Edition erscheint beim Laden einer Welt eine quadratische Pixelgrafik. Tatsächlich ist das eine grafische Darstellung der Chunks, die gerade generiert bzw. geladen werden. Jedes Pixel steht für einen Chunk. Die Farben stellen den Status des Chunks dar, wobei Weiß »fertig generiert« bedeutet.

Welten sichern, Welten kopieren

Nachdem du jetzt weißt, dass die Welt nicht auf einmal, sondern während deiner Bewegungen chunkweise gespeichert wird, wäre noch die Frage, wo diese Daten liegen. Die Chunks und weitere Spieldaten werden pro Welt in einem *Weltordner* gespeichert. Du findest ihn über das Einzelspieler-Menü: Wähle dort die gewünschte Welt (nicht öffnen), und klicke auf **Bearbeiten · Weltordner öffnen**. Ein neues Fenster zeigt den Weltordner.

Gehst du einen Ordner nach oben, siehst du den *saves*-Ordner, in dem alle Weltordner stehen. In der Bedrock Edition gibt es auch einen Weltordner, der aber aus dem Spiel heraus nicht erreichbar ist.



Im »saves«-Ordner stehen alle Weltordner (hier zwei).

Im Bearbeiten-Menü siehst du noch weitere Menüpunkte. Sie drehen sich um einen wichtigen Begriff: die *Sicherheitskopie*, auch *Backup* genannt.

Das Menü zur Bearbeitung einer Welt

Keiner wünscht es sich, aber manchmal passiert es doch: Der Computer stürzt ab oder geht anderweitig kaputt. Dann wäre es sehr schade, wenn du deine Minecraft-Welten nie wieder sehen könntest. Doch es gibt eine einfache Abhilfe: Immer wenn du in einer Welt viel verändert oder erreicht hast, machst du ein Backup.

So geht ein Backup mit der Java Edition:

- **Welt sichern**:
  - Gehe in das Einzelspieler-Menü, wähle die Welt aus (nicht spielen) und klicke auf **Bearbeiten · Sicherheitskopie erstellen**. Eine *zip*-Datei wird erzeugt.
  - Dann klicke im selben Menü auf **Sicherheitskopienordner öffnen** und kopiere die *zip*-Datei auf einen externen Speicher, z. B. auf einen USB-Stick.
- **Welt wiederherstellen**:
  - Starte den Launcher und wähle den Reiter **Installationen**. Fahre mit der Maus über die Liste und klicke bei der gewünschten Installation auf das Ordner-Icon. Ein neues Fenster zeigt den Ordner *.minecraft*.

  - Öffne die gesicherte *zip*-Datei. Kopiere den darin enthaltenen Weltordner in den Ordner *.minecraft\saves*.

So geht ein Backup mit der Bedrock Edition:

- **Welt sichern**:
  - Klicke im **Spielen**-Menü auf das Stift-Icon neben der Welt, um die Welt-Einstellungen zu öffnen.
  - Scrolle in den **Spiel**-Einstellungen (sie sind automatisch ausgewählt) ganz nach unten.
  - Klicke auf den Menüpunkt **Welt exportieren**. Jetzt kannst du einen beliebigen Speicherort wählen, z. B. einen USB-Stick. Eine *mcworld*-Datei wird erzeugt.
- **Welt wiederherstellen**:
  - Klicke im **Spielen**-Menü auf das Import-Icon rechts neben dem Menüpunkt **Neu erstellen**. Jetzt kannst du deine gesicherte Welt auswählen.
  - Schneller geht es, wenn du die *mcworld*-Datei doppelt anklickst. Dann startet automatisch die Bedrock Edition mit dieser Welt.

Weitere Menüpunkte beim Bearbeiten einer Welt

Im Menü zur Bearbeitung einer Welt gibt es noch weitere Menüpunkte.

Der Menüpunkt mit dem mysteriösen Namen **Symbol zurücksetzen** ändert das kleine Bild, das in der Liste der Einzelspielerwelten neben der Welt erscheint. Das funktioniert so:

1. Begib dich in der Welt an eine passende Stelle und schaue in die gewünschte Richtung. Schalte mit F1 die Anzeige deines Armes aus. Verlasse dann die Welt.
2. Setze mit dem Menüpunkt das Bild auf ein graues Standardbild zurück und speichere die Änderung.
3. Betritt die Welt erneut und warte etwa zehn Sekunden. Dadurch entsteht ein neues Bild vom ersten Blick auf die Welt. Jetzt kannst du F1 wieder einschalten.

Sollte das nicht richtig funktionieren, kannst du auch den Weltordner öffnen und dort die Datei *icon.png* gegen ein quadratisches Bild austauschen.

Eine Einzelspielerwelt mit »Symbol« (das kleine Bild)

Ebenso mysteriös ist der Menüpunkt **Welt optimieren**. Wenn die Welt schon älter ist und bereits neue Minecraft-Versionen erschienen sind, bringt dieser Menüpunkt die Weltdaten auf den aktuellen Stand. Normalerweise brauchst du diesen Menüpunkt nicht, weil das Aktualisieren der Daten beim Laden von Chunks während des Spiels geschieht. Aber manchmal führt das zu störenden Rucklern beim Spielen. Daher kannst du mit diesem Menüpunkt die Aktualisierung einmalig für die ganze Welt durchführen. Das kann dann, je nach Größe der Welt, einige Zeit dauern.

Du weißt jetzt, wie du eine Welt sicherst und wieder herstellst. Damit hast du aber nicht nur ein Backup für den Notfall, sondern auch die Möglichkeit, deine Welt an einen anderen Ort zu bringen. Das eröffnet neue Möglichkeiten!

Sichere die Welt z. B. auf einen USB-Stick. Damit gehst du zu deinen Freunden, die ebenfalls Minecraft haben, und stellst die Welt dort wieder her. Nun könnt ihr gemeinsam darin spielen, wenn ihr exakt dieselbe Minecraft-Version habt. Dazu betritt ein Spieler die Welt und stellt sie über das Spielmenü (Esc) den anderen zum Mitspielen zur Verfügung. In der Java Edition heißt der Menüpunkt **Im LAN öffnen**, in der Bedrock Edition **Ins Spiel einladen**.

Dann sehen die Mitspieler, die sich über LAN-Kabel oder WLAN mit dem Heimnetzwerk verbunden haben, die Welt in ihrem Mehrspieler-Menü.

Links: In der Java Edition erscheinen LAN-Welten ganz unten im Mehrspieler-Menü.
Rechts: In der Bedrock Edition sind sie im Menüpunkt »Spielen« unter »Freunde« zu sehen.

Nach dem Ende des Spiels sicherst du den aktuellen Zustand zurück auf deinen USB-Stick, stellst ihn zu Hause auf deinem Computer wieder her und spielst damit weiter. Deine Freunde können natürlich mit ihrer Kopie auch weiterspielen, falls ihr sie nicht gelöscht habt. Dann gibt es zwei Welten, die sich unterschiedlich weiterentwickeln.

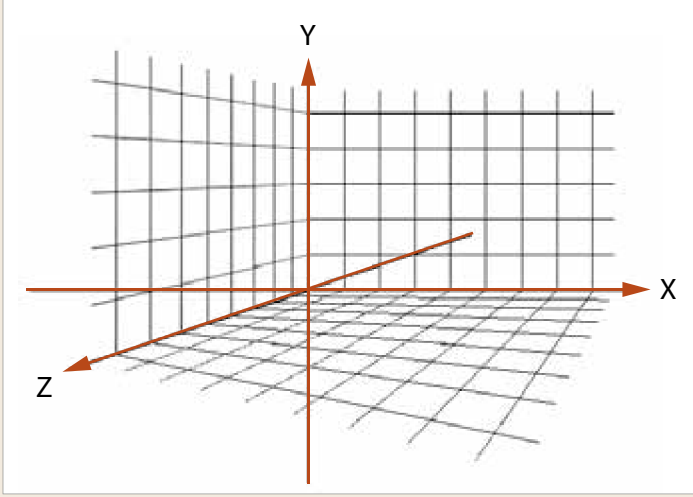
RAUM IST EINE ILLUSION – ES GIBT KEINE BLÖCKE

Gerade hast du erfahren, dass die aktive Welt gar nicht so riesig ist, wie du immer dachtest, sondern hinter dem Horizont endet. Und nun soll es auch keine Blöcke geben? Schwer zu glauben, denn du kannst sie sehen und sogar hören und spüren. Besonders wenn du das falsche Werkzeug benutzt, spürst du, dass das Abbauen richtig lange dauert, du hörst Geräusche, und kleine Partikel fallen zu Boden.

Baust du einen Block mit dem falschen Werkzeug ab, dauert es besonders lange.

Blöcke sehen

Aber für das Spiel besteht die Welt nicht aus Blöcken, sondern nur aus *XYZ-Koordinaten*, die du dir als dreidimensionales Gitter vorstellen kannst. Jeder Kreuzungspunkt in diesem Gitter stellt einen Block dar.

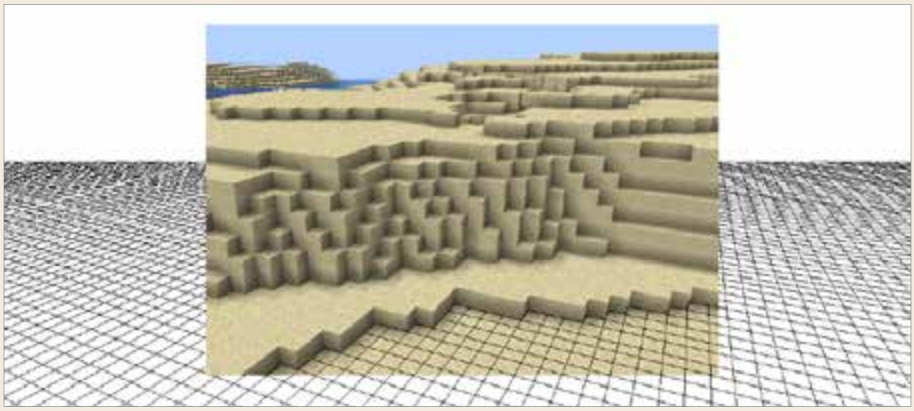


Das Blockgitter: Die X-Koordinate steigt nach Osten, Y nach oben und Z nach Süden.

Um die Blöcke unterscheiden zu können, hat jede Blockart eine *Identifikation* (ID), z. B. `minecraft:stone` für den Stein und `minecraft:torch` für die Fackel. Intern rechnet das Spiel nur mit XYZ-Koordinaten und IDs.

Deine Spielfigur befindet sich irgendwo in diesem Gitter. Die Fußposition wird durch XYZ-Koordinaten bestimmt. Dazu kommen deine *Blickrichtung* und deine *Kopfneigung*, d. h., ob du nach oben oder unten schaust. Außerdem kannst du in den Minecraft-Optionen unter **Sichtfeld** einstellen, wie weit du aus den Augenwinkeln siehst (*field of view*, FOV). Mit diesen Informationen berechnet das Spiel, auf welchen Ausschnitt des Koordinatengitters du gerade schaust und welche Block-IDs dort liegen. Daraus erzeugt das Spiel ein Bild. Dieser Vorgang wird **Rendern** genannt.

Du bist also nur ein Punkt in einem Koordinatengitter. Bewegst du dich in eine Richtung oder drehst deinen Kopf, muss ein neues Bild gerendert werden. Was hinter deinem Rücken liegt, interessiert das Rendern nicht. Mit anderen Worten: Deine Minecraft-Augen sind wie ein Suchscheinwerfer: Nur wo du hinschaust, entsteht ein Bild, überall sonst gibt es nur ein farbloses Gitter aus Block-IDs, das du niemals sehen kannst.



Die Welt ist nur sichtbar, wo du hinschaust.

Aber auch wenn du dich nicht bewegst, muss das Spiel in deinem Sichtfeld ständig viele Bilder rendern, damit du z. B. siehst, wie sich andere Lebewesen bewegen. Doch wie viele Bilder pro Sekunde muss das Spiel eigentlich rendern?

Ab etwa 16 Bildern pro Sekunde nimmt das Auge eine Bewegung wahr, wie bei einem Daumenkino. Kinofilme zeigen 24 Bilder pro Sekunde. Für Computerspiele sind mindestens 60 Bilder pro Sekunde üblich. Das wird *Bildrate* genannt, die in *fps* (*frames per second*) gemessen wird. Sinkt die Bildrate unter 30 fps, erscheinen die Bewegungen im Spiel abgehackt.

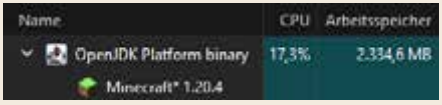
Hast du einen leistungsfähigen Computer, kann die Anzahl der gerenderten Bilder pro Sekunde sehr hoch sein. Doch es gibt eine technische Grenze, die *Bildwiederholrate* deines Monitors. Das ist die Anzahl der Bilder, die dein Monitor pro Sekunde anzeigt, sie wird in Hertz (Hz) gemessen. Hast du einen Monitor mit 60 Hz, zeigt er 60 Bilder pro Sekunde, bei 144 Hz sind es 144 Bilder pro Sekunde. Das Rendern kann deutlich mehr Bilder erzeugen, z. B. 250 fps. Die Grafikeinstellung **VSync** (*video synchronisation*) begrenzt das Rendern auf die Bildwiederholrate deines Monitors, denn das Rendern von mehr Bildern pro Sekunde ist überflüssige Rechenarbeit. Manchmal erzeugt diese Begrenzung jedoch störende Bildeffekte. Dann kannst du ausprobieren, ob das Ausschalten von VSync etwas bringt.

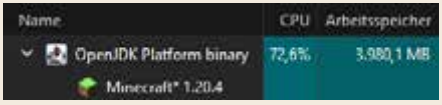
Das Rendern eines einzigen Bildes kann bei Animationsfilmen durchaus ziemlich lange dauern, aber das macht nichts, weil der Film erst vorgeführt wird, wenn alles fertig gerendert ist. Minecraft dagegen ist ein Echtzeitspiel, das Rendern muss hier während des laufenden Spiels geschehen und deshalb ex-

trem schnell ablaufen. Daher berechnet das Spiel vor dem eigentlichen Rendern zuerst einmal, welche Blockseiten du überhaupt sehen kannst. Rückseiten (auch bei Glas) und verdeckte Blöcke werden grundsätzlich weggelassen. Das bedeutet, das gerenderte Bild ist nur eine Kulisse! Alle Blöcke sind hinten offen. Rund um dich herum siehst du nur eine Oberfläche aus bunten Blockkulissen, dahinter ist alles leer.

Auch beim durchsichtigen Glas werden die Rückseiten nicht gerendert. Eigentlich müsste der hellblaue Rahmen auf allen sechs Seiten zu sehen sein.

Zusätze wie HD-Texturenpakete oder Shader, die in den nächsten Kapiteln vorgestellt werden, können das Rendern erheblich verlangsamen. Abhilfe schafft eine *Grafikkarte*, die viele Renderaufgaben übernimmt und damit den Computer entlastet. Außerdem solltest du möglichst wenig Programme parallel zu Minecraft laufen lassen. Unter Windows öffnet die Tastenkombination Strg + ⇧ + Esc den *Task-Manager*, der dir anzeigt, welche Prozesse besonders viel Rechenleistung (CPU) und Arbeitsspeicher (RAM) beanspruchen.

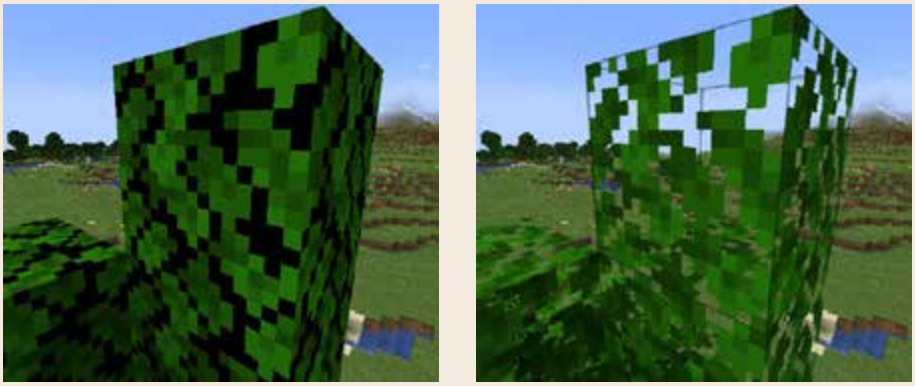




Links: Die Java Edition im Task Manager bei 2 Chunks Sichtweite und keiner Bewegung. Rechts: Ohne Grafikkarte belastet schnelle Bewegung im Spiel die CPU mit Renderaufgaben. Zusätzlich lädt eine hohe Sichtweite mehr Chunks und füllt den Arbeitsspeicher.

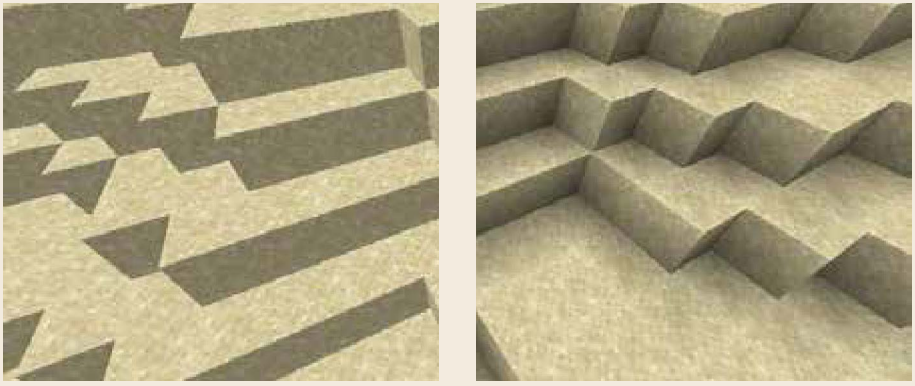
Ist dein Spiel langsam, obwohl du schon deine Sichtweite verringert und andere Programme beendet hast, gibt es noch weitere Grafikeinstellungen, die

das Rendern entlasten. Da wäre z. B. der **Grafikmodus**, der vor allem die Darstellung von Laub beeinflusst. Normalerweise hat Laub viele kleine Löcher (Einstellung **Schön**), aber die Berechnung der dahinter sichtbaren Blöcke ist besonders aufwendig. Ohne Löcher (Einstellung **Schnell**) wird das Rendern entlastet.



Laub in den Einstellungen »schnell« und »schön«

Das größte Problem fürs Rendern ist jedoch die **Beleuchtung**. Eine realistische Berechnung von Licht, Schatten und Wasserspiegelungen ist sehr aufwendig. Damit das Spiel nicht belastet wird, verzichtet Minecraft darauf und wendet einen Trick an: Jeder Block erhält einen weichen Schatten, egal, woher das Licht kommt. Das ist zwar völlig unrealistisch, reicht aber für einen erstaunlichen Räumlichkeitseffekt. Weil diese Berechnung trotzdem noch einiges an Rechenleistung erfordert, kannst du sie in den Grafikeinstellungen unter **Weiche Beleuchtung** ausschalten.



Links ohne, rechts mit der Einstellung »Weiche Beleuchtung«

Ein aufgemalter Schatten auf den Blöcken reicht aber nicht in Kellern und Höhlen, da soll es richtig dunkel sein. Damit ein realistischer Übergang zwischen Hell und Dunkel entsteht, gibt es 16 *Helligkeitsstufen* von 15 (volle Helligkeit) bis 0 (maximale Dunkelheit). Dazu ermittelt das Spiel für jeden Block, wie weit er von freiem Himmel entfernt ist. Je größer die Entfernung, desto dunkler wird er gerendert, bis er nach 15 Blöcken Entfernung vom Himmel in maximaler Dunkelheit liegt.

Schatten in einer Wüstenhöhle. Du siehst die verschiedenen Helligkeitsstufen besonders deutlich, weil die weiche Beleuchtung ausgeschaltet ist.

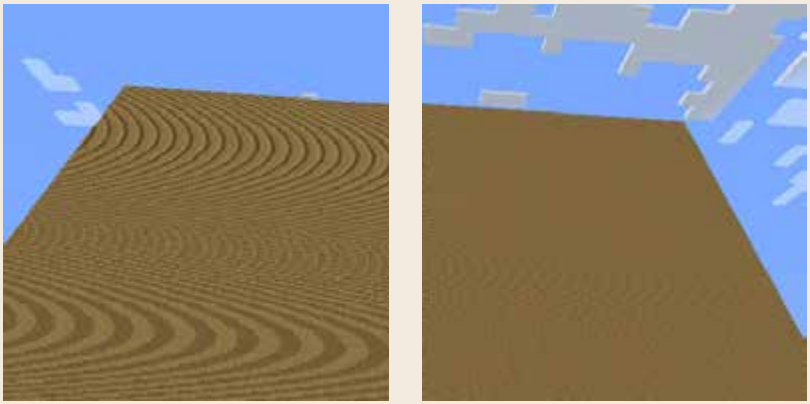
Neben dem *Himmelslicht*, das auf der Oberfläche überall mit gleicher Stärke 15 vorhanden ist, gibt es noch künstliche Lichtquellen wie Fackeln, Laternen und Leuchtflechten (*Blocklicht*), die ihre Umgebung nur in einem kleinen Umkreis beleuchten. Jede leuchtende Blockart hat ihre eigene Lichtstärke zwischen 15 (z. B. Feuer, Laterne) und 1 (z. B. brauner Pilz). Die Lichtstärke gibt an, wie weit das Licht des Blocks reicht, wobei sie mit jedem Block Entfernung um 1 abnimmt. Eine Laterne beleuchtet ihren eigenen Block und andere Blöcke 14 Blöcke weit. Eine weitere Vereinfachung des Spiels: Lichtstärken addieren sich nicht, d. h., zwei Fackeln sind genauso hell wie eine.

Das Spiel muss für jeden Block berechnen, wie viel Licht ihn von anderen Blöcken (z. B. Fackeln) trifft. Auch bei diesem Bild ist die weiche Beleuchtung ausgeschaltet.

Im Debug-Bildschirm zeigt die Zeile **Client Light** die Helligkeitsstufe an deiner Position an, wobei **sky** für das Himmelslicht steht und **block** für die hellste künstliche Lichtquelle in deiner Nähe, die dich trifft.

Jetzt verstehst du langsam, warum das Rendern so aufwendig ist: Das Spiel muss den sichtbaren Ausschnitt im Blockgitter berechnen, die darin enthaltenen Block-IDs sammeln, die sichtbaren Seiten der Blöcke ermitteln, die zugehörigen Blocktexturen laden und perspektivisch korrekt verzerren, die weiche Beleuchtung verteilen, die Helligkeitsstufe für jeden sichtbaren Block berechnen und jede Blocktextur entsprechend abdunkeln. Dazu kommen alle möglichen Kreaturen, die durchs Bild laufen, gedroppte Gegenstände, die auf dem Boden liegen, herumfliegende Partikel und einiges mehr. Und all das wird im Idealfall über 100-mal pro Sekunde neu berechnet.

Zusätzlich zu einer möglichst realistischen Darstellung hat das Rendern noch mit **optischen Störungen** zu kämpfen: Durch das Pixelraster des Monitors in Verbindung mit den Pixeln der Blockoberflächen entstehen sogenannte *Moiré-Effekte*. Besonders deutlich wird das beim Betrachten von großen Flächen aus der Entfernung. Um diese Effekte zu verhindern, hat das Spiel jede Blockoberfläche in verschiedenen Verkleinerungen gespeichert. Die Verkleinerungen haben weniger Pixel, wodurch der Moiré-Effekt nicht auftritt. Das wird *Mip-Mapping* genannt. Das bewegte Spiel wird dadurch realistischer, aber Screenshots sehen in der Vergrößerung im Hintergrund verwaschen aus.



Eine große Bretterwand, links ohne, rechts mit Mip-Mapping Stufe 2

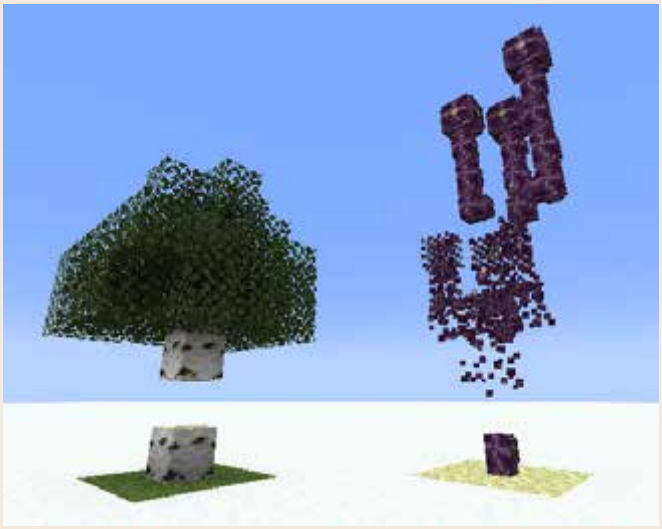
Blöcke hören und spüren

Während du dich im Koordinatengitter bewegst, sorgt das Rendern für die Illusion einer blockigen Landschaft. Aber realistisch wird das Ganze erst, wenn du die Blöcke auch hören und spüren kannst. Das verstärkt die sogenannte *Immersion*, den Effekt, dass dein Gehirn die Illusion des Spiels für real hält. Wie gut das funktioniert, weißt du, wenn du dich an die vielen aufregenden Momente in deiner Minecraft-Welt erinnerst.

Die **Blockgeräusche** sind dabei ein wichtiges Element. Unterschiedliche Blöcke machen beim Platzieren, Darüberlaufen und Abbauen unterschiedliche Geräusche.

1. Stelle die Minecraft-Option **Musik & Geräusche · Blöcke** auf 100 %, und stelle deinen Lautsprecher etwas lauter.
2. Setze dann Honigblöcke, Schnee, Laub, Wolle, Sand, Glas und Kies auf den Boden, und achte auf die unterschiedlichen Geräusche. Laufe auf diesen Blöcken, und baue sie wieder ab, um auch diese Geräusche zu hören.

Neben den Geräuschen verstärken physikalische Effekte die Illusion. Zu einer realistischen Physik gehört vor allem **Schwerkraft**, aber die ist aufwendig zu berechnen. In Minecraft gibt es nur wenige Blöcke mit Schwerkraft, vor allem Sand, Kies und Trockenbeton. Die anderen Blöcke schweben einfach in der Luft, wenn du ihren Untergrund entfernst. Das ist besonders kurios beim Fällen von Bäumen. Die Choruspflanze ist eins der wenigen Gewächse, das beim Fällen in sich zusammenbricht – allerdings nur als Partikelwolke.

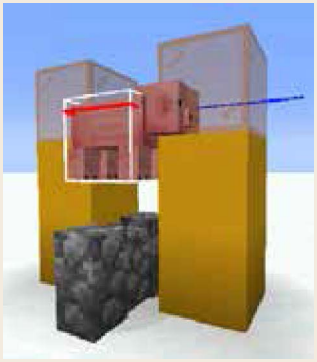


Entfernung des zweiten Blocks bei einem Baum und einer Choruspflanze.
Der Baum schwebt in der Luft, die Choruspflanze zerfällt in einer Kettenreaktion

Wesentlich einfacher als die Realisierung von Schwerkraft ist die Berechnung von **Kollisionen**. Blöcke, Lebewesen und sonstige Objekte haben eine *Kollisionsbox*. Das ist ein unsichtbarer kastenförmiger Bereich um die Figur. Sobald das Spiel eine Berührung zweier Kollisionsboxen berechnet, stoppt es die Bewegungen, was dir wie ein Anstoßen erscheint.

In den meisten Fällen ist die Kollisionsbox identisch mit der *Hitbox*. Das ist der Bereich, in dem du den Block oder das Lebewesen treffen kannst. Manchmal unterscheiden sich die beiden Boxen, das kannst du leicht mit Pfeil und Bogen feststellen:

- Bei **Blöcken** siehst du die Hitbox, wenn du sie mit dem Fadenkreuz anschaust. Mauern haben eine unsichtbare Kollisionsbox, die ausnahmsweise höher ist: Du kannst einen Pfeil knapp über die Mauer schießen (Hitbox verfehlt), aber du kannst nicht auf sie springen (Kollisionsbox getroffen).
- Bei **Objekten** siehst du die Box mit F3 + B. Das Spiel nennt sie »Hitbox«, aber genau genommen ist es die Kollisionsbox. Beweis: Ein Schwein passt mit seiner würfelförmigen Box genau in ein 1×1-Loch, der große Kopf ragt in die Wand (keine Kollision). Die echte Hitbox ist dagegen größer als die Anzeige, denn ein Pfeil trifft das Tier sehr wohl am Kopf. Das Spiel zeigt mit der Box außerdem einen blauen Strahl für die Blickrichtung. Lebewesen haben zusätzlich einen roten Ring in Augenhöhe.



Links: Leitern haben eine größere Box als Ranken, daher kannst du auf Leitern stehen, auf Ranken nicht. Rechts: Das Schwein hat eine würfelförmige Kollisionsbox ohne den Kopf, daher passt es zwischen zwei Blöcke. Das Tier steht auf der unsichtbaren Kollisionsbox der Mauer, die höher ist als ein Block, damit man nicht darüberspringen kann.

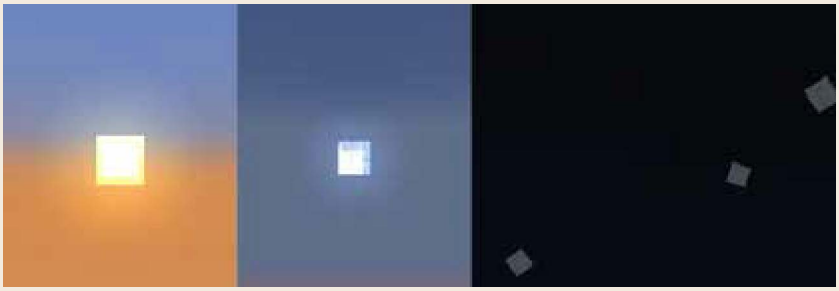
In Minecraft gibt es den sogenannten **Zuschauermodus**, der für Minispiele gedacht ist: Zuschauer oder Spieler, die aus einem Minispiel ausgeschieden sind, sollen das weitere Geschehen beobachten können, ohne zu stören. Im Zuschauermodus sind sie unsichtbar und stoßen nirgends an. Sie können keine anderen Spieler und auch sonst keine Lebewesen schubsen und keine Gegenstände aufheben. Erreicht wird das durch das Abschalten der Kollisionsberechnung. Ohne die Kollisionsberechnung kannst du im Zuschauermodus auch durch Wände fliegen und hinter die Kulissen schauen.

Fliegst du im Zuschauermodus unter die Erde, siehst du, dass die Oberfläche nur eine Kulisse ist, denn das Spiel rendert nur die Vorderseiten der Blöcke, und auch nur, wenn sie nicht verdeckt sind. Die anderen Blockseiten werden weggelassen.

In der Java Edition schaltest du den Zuschauermodus mit F3 + N ein und aus. In der Bedrock Edition öffnest du mit der Taste T die Befehlseingabe und gibst den Befehl `/gamemode spectator` ein. Zurück kommst du mit dem Befehl `/gamemode survival` oder `/gamemode creative`.

ZEIT IST EINE ILLUSION – ES GIBT WEDER TAG NOCH NACHT

Die Zeit ist in Minecraft allgegenwärtig: Sonne, Mond und Sterne wandern langsam über den Himmel, Ackerpflanzen und Bäume wachsen, Laub zerfällt, fallengelassene Gegenstände verschwinden, Schnee und Eis schmelzen, Jungtiere werden erwachsen etc.



Sonne, Mond und Sterne. Je nach Uhrzeit ist der Himmel unterschiedlich gefärbt. In der Java Edition siehst du die Sterne mit dem Fernrohr besser, wenn du F1 drückst.

In Wirklichkeit sind Tag und Nacht nur eine Illusion. Hast du schon einmal versucht, bis zum Minecraft-Mond zu fliegen, vielleicht um das Ende zu erreichen? Das kann nicht funktionieren, denn Sonne, Mond und Sterne sind nur kleine Bildchen (*Sprites*), die langsam hinter den gerenderten Blöcken bewegt werden. Eine interne Uhr steuert ihre Bewegung und ändert langsam die Himmelsfarbe, während die Stärke des Himmelslichts immer gleich bleibt. Mit F3 siehst du in der Java Edition, dass unter freiem Himmel der Wert **Client Light: sky** immer auf 15 steht, egal ob bei Tag oder Nacht. Die Nacht ist nur eine optische Illusion: Der Himmel und die Blöcke werden einfach dunkler gerendert.

Mit der Zeit verhält es sich in Minecraft genauso wie mit der Welt: Sie ist nur da, wo du bist. Hinter dem Horizont gibt es keine Welt und keine Zeit. Deshalb

musst du immer in der Nähe bleiben, wenn du auf das Wachsen von Pflanzen, das Schlüpfen von Schildkröten oder das Auswachsen von Jungtieren wartest. Sie können nur im geladenen Bereich der Welt wachsen. Probiere das einmal aus:

1. Stelle deine Sichtweite auf den Minimalwert 2 und gehe in eine Einzelspielerwelt.
2. Platziere dort einige Eisblöcke und stelle Fackeln dazu.
3. Merke dir den Ort und fliege drei Chunks (etwa 50 Blöcke) weit weg. Der Ort mit dem Eis liegt dann außerhalb deiner Sichtweite und wird aus dem Spiel genommen.
4. Platziere am neuen Ort ebenfalls Eis mit Fackeln und warte, bis es geschmolzen ist.
5. Fliege dann zurück. Der erste Ort wird wieder geladen. Das Eis ist noch da, weil die Zeit stillstand, während du woanders warst. Wenn du also Ackerbau oder Viehzucht betreibst oder auf den Ofen wartest, solltest du auf deine eingestellte Sichtweite achten. Gehst du weiter weg, wird der Bereich hinter dem Horizont aus dem Spiel genommen, und die Zeit bleibt dort stehen.

So tickt die Zeit

In Computerspielen, die mit *Spielrunden* funktionieren, ändert sich die Zeit nur, wenn du einen Spielzug machst. Typische Beispiele sind Kartenspiele und Strategiespiele. Solange du in Ruhe überlegst, ändert sich am Spiel nichts. Erst deine Eingabe startet die Berechnungen für eine neue Runde und bewirkt einen Spielfortschritt.

Im Gegensatz dazu ist Minecraft ein *Echtzeitspiel*. Hier läuft die Zeit weiter, auch wenn du nichts tust. Ein internes, rhythmisches Signal sorgt für den Spielfortschritt wie eine tickende Uhr. Dieses Signal wird **Game-Tick** oder kurz *Tick* genannt. Mit jedem Tick bewegt das Spiel alle Lebewesen an eine neue Position. Damit das wie eine flüssige Bewegung aussieht, ist das Zeitintervall sehr kurz. Ein Tick erfolgt alle 50 Millisekunden, d h. 20-mal pro Sekunde.

Du siehst nur ein paar Schweine, aber es sind fast 300 bewegliche Objekte in den geladenen Chunks, deren Positionen das Spiel 20-mal pro Sekunde neu berechnet.

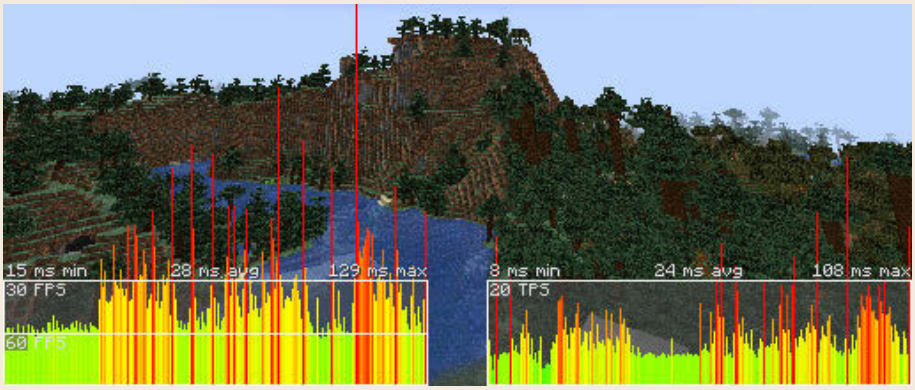
Das Berechnen neuer Positionen ist aber nur ein winziger Teil der vielen Aktionen, die bei jedem einzelnen Tick durchgeführt werden. Das Spiel verändert mit jedem Tick z. B. auch sämtliche Redstone-Schaltkreise und fließende Flüssigkeiten und reagiert auf deine Aktionen. Wenn du einen schnellen Computer hast, ist alles erledigt, bevor der nächste Tick kommt. Dann tickt das Spiel mit einer stabilen Rate von 20 Ticks pro Sekunde (*tps*). Bei einem langsamen oder überlasteten Computer kann die Abarbeitung der Aktionen länger als 50 Millisekunden dauern. Dann schafft das Spiel keine 20 tps mehr und überspringt ein oder mehrere Ticks. In diesem Fall bewegen sich Lebewesen und andere Spieler nicht mehr flüssig, sondern ruckartig. Diesen unschönen Effekt nennt man *Lag* (deutsch »Verzögerung«).

Besonders die Betreiber von Multiplayer-Servern möchten Lags vermeiden und beobachten kritisch ihre tps-Rate. Liegt der Wert dauerhaft unter 20 tps, muss ein Serverbetreiber entweder einen leistungsfähigeren Server mieten oder die Belastung des Spiels einschränken, indem er beispielsweise die Anzahl der Tiere begrenzt.

Mit der Java Edition kannst du sogar das schlagende Herz des Spiels sehen! Drücke dazu die Tastenkombination F3 + 2. Dann siehst du unten zwei Diagramme:

- Links die **Bildrate** in fps (Bilder pro Sekunde). Sinkt die Bildrate, steigen die Striche und ändern ihre Farbe von Grün zu Rot. Weiße Querstriche markieren die 60- und die 30-fps-Grenze. 60 fps sind okay, ab 30 fps beginnen Lags.
- Rechts die **Tickrate** in tps (Ticks pro Sekunde). 20 tps sind optimal. Sinkt die Tickrate, steigen auch hier die Striche und ändern ihre Farbe.
- Durch erneutes Drücken von F3 schaltest du die Anzeige wieder aus. Im Mehrspielermodus siehst du dieses Diagramm nicht.

Du erkennst die Diagramme besser, wenn du in den Grafik-Einstellungen die **GUI-Grösse** auf **1** stellst.



In diesem Beispiel ist die Sichtweite sehr hoch eingestellt, und es gibt keine Grafikkarte. Folge: Das Spiel ist mit Rendern überlastet.

Die Diagramme machen auch deutlich: Das Rendern geschieht unabhängig von den Ticks. Die Ticks bestimmen das Voranschreiten der Zeit im Spiel, das Rendern hat damit nichts zu tun. Während es nie mehr als 20 Ticks pro Sekunde geben kann, muss das Rendern deutlich öfter geschehen – 60 Mal pro Sekunde oder mehr –, damit du bei schnellen Mausbewegungen ein flüssiges Spiel siehst.

Um einen Eindruck zu bekommen, wie Lags wirken, mache folgendes Experiment:

1. Schalte die Diagramme mit F3 + 2 ein.
2. Stelle deine Sichtweite auf 2 Chunks und fliege durch die Welt. Die Diagramme sollten grün bleiben, du hast über 60 fps.

3. Erhöhe deine Sichtweite stark. Ohne dass du dich bewegst, siehst du schon gelbe, orange und rote Striche in den Diagrammen.
4. Nach einiger Zeit sind alle Chunks generiert bzw. geladen und die Diagramme normalisieren sich wieder.
5. Sobald du durch die Welt fliegst, geraten sie aber wieder in den roten Bereich, und die fps-Anzeige sinkt unter 60 – besonders wenn du gleichzeitig schnell die Maus hin und her bewegst und damit ständig den Renderausschnitt änderst.
6. Schaue zum Himmel, wo es nichts zu rendern gibt, die fps-Werte verbessern sich deutlich.

Falls bei diesem Experiment alles im grünen Bereich bleibt, kann man dir gratulieren, dann hast du eine gute Grafikkarte oder einen besonders leistungsfähigen Computer.

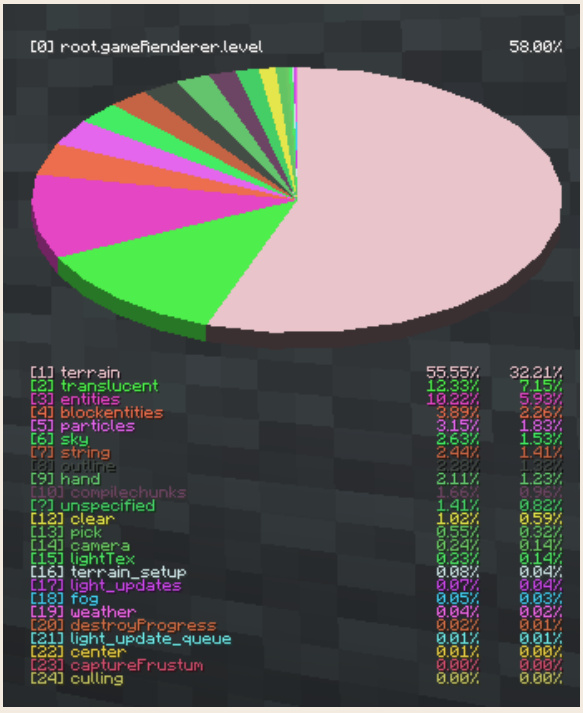
Bei der Umstellung auf eine sehr große Sichtweite zeigen nicht nur die Diagramme die beginnende Belastung an, du kannst auch das verlangsamte Rendern der Chunks am Horizont beobachten.

Spielst du im Mehrspielermodus auf einem Minecraft-Server, interessiert dich die Verbindungsqualität (*Ping*). Mit F3 + 3 siehst du dazu zwei Diagramme, links die Datenmenge, die pro Sekunde vom Server zu dir übertragen wird, rechts den Ping (je kleiner, desto besser).

Für Spiel-Entwickler und Mod-Entwickler ist das fps-Diagramm noch zu ungenau. Sie möchten wissen, *welche Aktionen* innerhalb eines Ticks am meisten

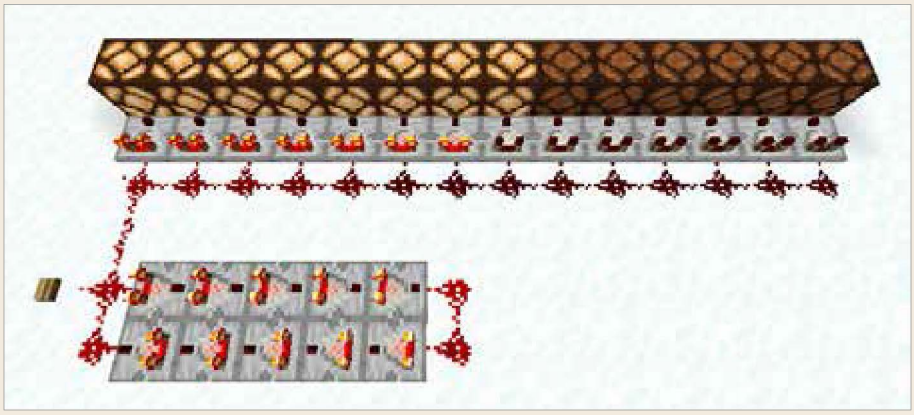
Rechenzeit kosten, damit sie dort eingreifen und die Programmierung optimieren können. Dafür gibt es in der Java Edition eine weitere Anzeige, den *Profiler*, den du mit F3 + 1 öffnest.

Unter dem Tortendiagramm siehst du die internen Funktionen des Spiels, die aktuell am meisten Rechenzeit kosten. Tippst du die zugehörige Zahl ein, wechselt das Diagramm in eine Detailansicht. Mit 0 kommst du wieder zurück. Die Bezeichnung »unspecified« ist der Rest zu 100 %, dort gibt es keine weiteren Funktionen, daher steht dort auch keine Zahl.



Dieses Profiler-Beispiel zeigt das Rendern (»root.gameRenderer.level«). Du erkennst: Am meisten Rechenzeit kostet das Rendern der Oberfläche (55,55 % der Renderzeit bzw. 32,21 % der Gesamtzeit). Auf Platz zwei und drei folgen durchsichtige Blöcke und bewegliche Objekte.

Besonders deutlich wird die Existenz des Ticks bei Redstone-Schaltungen. Redstone-Signale fließen nicht auf einen Schlag durch eine komplette Schaltung, sondern schrittweise. Jedes Redstone-Bauteil verzögert die Weiterleitung des Signals um 2 Ticks oder 0,1 Sekunden. Diese Einheit nennen Spieler **Redstone-Tick** und berechnen, wie viele Redstone-Ticks ein Signal bis zum Erreichen des Ziels braucht.



Das Signal durchläuft unten zehn Redstone-Komparatoren und braucht dafür zehn Redstone-Ticks oder eine Sekunde. Bei jedem Durchlauf wird das Signal zwei Stufen schwächer. Nach sieben Sekunden ist es so schwach, dass es die Redstone-Lampen nicht mehr erreicht.

Der Beobachter erzeugt ein sehr kurzes Signal, es dauert nur ein Redstone-Tick. In der Java Edition kann der klebrige Kolben einen Block so schnell nicht wieder zurückziehen, er reißt ab.

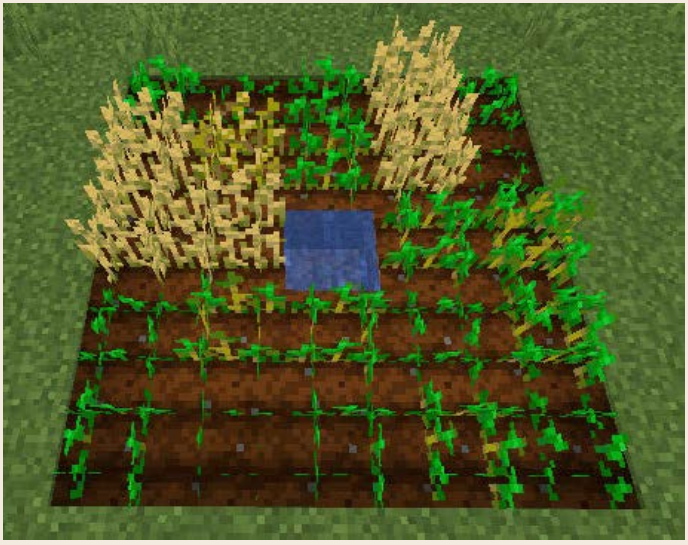
Das Zeitgewitter

Trotz der rasend schnell tickenden internen Uhr gibt es in Minecraft viele Vorgänge, die nur langsam voranschreiten, sie brauchen Zeit. Das kennst du z. B. vom Erzschmelzen in einem Ofen oder vom Brauen. Auch die Ausbreitung von Wasser oder – noch deutlicher – Lava braucht eine bestimmte Zeit. Startet ein Block eine Aktion mit **fester Zeitdauer**, trägt er das Ende in eine interne *Zeittabelle* ein. Mit jedem Tick prüft das Spiel alle Zeiten in dieser Tabelle. Ist eine Zeit erreicht, ändert es den Zustand des entsprechenden Blocks. So erlischt beispielsweise ein brennender Ofen nach einer ganz bestimmten Zeit.

Neben den wenigen Blöcken mit regelmäßigem Zeitablauf gibt es jedoch sehr viel mehr Blöcke mit **zufälliger Veränderung**. Dazu gehört besonders das Pflanzenwachstum (Ackerpflanzen, Ranken, Bäume), aber auch zerfallendes Laub (wenn du den Stamm entfernst) und schmelzendes Eis neben einer Fackel. Das Spiel wäre völlig überlastet, wenn sich jeder einzelne Pflanzen-, Laub- und Eisblock in die Zeittabelle eintragen würde. Stattdessen erzeugt das Spiel ein ständiges, unsichtbares Zeitgewitter um dich herum, das auf alle Blöcke einprasselt: Pro Tick, also 20-mal pro Sekunde, werden in jedem Chunk, der in deiner Sichtweite liegt, über die gesamte Welthöhe 72 zufällig ausgewählte Blöcke angestoßen. Das wird *Block-Tick* genannt. 72 Blöcke klingt wenig, aber weil das gleichzeitig in vielen Chunks um dich herum passiert, ergibt das ein Gewitter von über 280.000 Block-Ticks pro Sekunde!

Die meisten Blöcke (z. B. Stein) reagieren nicht auf den Block-Tick. Aber wenn zufällig ein Block getroffen wird, der sich verändern kann (z. B. ein Pflanzenblock), erhält er ein **Block-Update**. So nennt man die Änderung eines Blocks durch das Spiel: Eis wird zu Wasser, Setzling wird zu Baum etc.

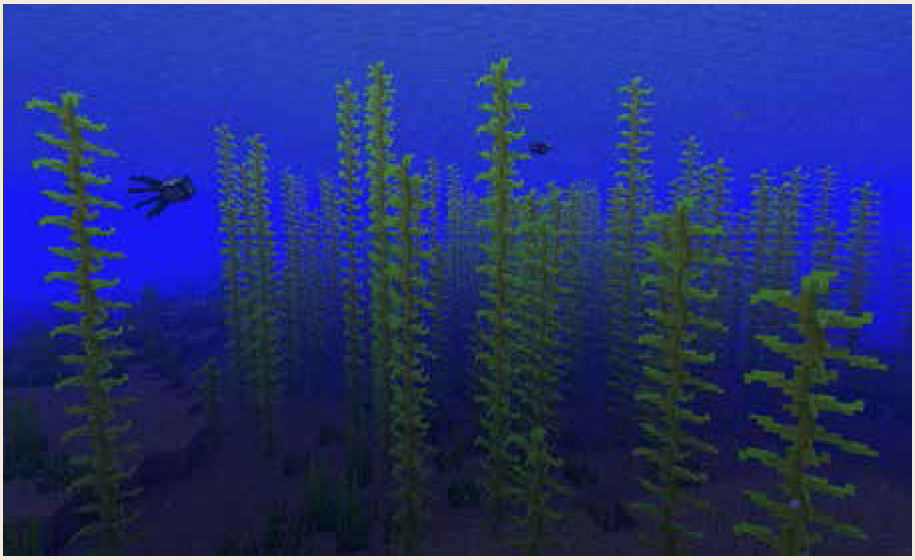
Weil die meisten Blöcke aus Stein sind, gibt es trotz vieler Tausend Block-Ticks nur wenige zufällige Block-Updates. Dadurch entsteht die Illusion, dass um dich herum alle Pflanzen ein individuelles Eigenleben führen und sich langsam und unregelmäßig verändern.



An einem Weizenfeld kannst du besonders gut die zufällig ausgeführten Block-Updates erkennen.

Die meisten Pflanzen wachsen dabei nicht wirklich, die Blöcke ändern nur ihr Aussehen. Bei Weizen siehst du das besonders deutlich: Diesen Block gibt es in acht Zuständen, die in ihrer Abfolge das Wachstum simulieren. Auch bei den Bäumen gibt es kein Wachstum. Sobald ein Setzling sein zufälliges Block-Update erhält, wird sofort der fertige Baum generiert, falls genug Platz da ist. Es gibt keine Übergangsstufen, Bäume sind schlagartig einfach da. Du kannst das selbst erleben, wenn du einen Setzling mit Knochenmehl düngst. Mehr kann das Spiel nicht. Ein fertiger Baum wächst nicht weiter.

Daneben gibt es auch einige Gewächse, die tatsächlich blockweise größer werden: Kakteen, Zuckerrohr, Bambus, Seetang, diverse Ranken und die Choruspflanze, die sich eindrucksvoll verzweigt. Hier muss das Spiel dafür sorgen, dass die Pflanzen nicht zu groß werden. Kakteen und Zuckerrohr können nur maximal drei Blöcke hoch werden. Bei den anderen Pflanzen hat die Spitze eine Stopp-Eigenschaft. Sobald diese per Block-Update gesetzt wird, wächst die Pflanze nicht weiter. Nur normale Ranken wachsen immer weiter und weiter, sofern sie Platz haben.



Seetang wächst blockweise nach oben, hat aber eine zufällige Stopp-Eigenschaft, damit die Pflanzen unterschiedlich hoch werden.

Zufällige Block-Updates sorgen aber nicht nur für ein langsames Pflanzenwachstum, sondern insgesamt für eine sich langsam verändernde Natur, wie diese Beispiele zeigen:

- Ackerboden wird in Wassernähe nass oder ohne Bepflanzung zu Erde.
- Grasblöcke und Myzel breiten sich auf Erde aus oder werden bei Bedeckung wieder zu Erde.
- Pilze breiten sich auf andere Blöcke aus.
- Generiertes Laub ohne Stamm zerfällt (von dir selbst gesetztes Laub zerfällt nicht, daher kannst du aus Laub problemlos Hecken bauen).
- Schildkröten- und Schnüfflereier verändern sich oder erzeugen ein Jungtier.
- Feuer breitet sich aus oder erlischt.
- Lava erzeugt ein Feuer in seiner Nähe. Es sind tatsächlich Block-Updates, die für das Feuer sorgen, und nicht die wegspringenden Funkenpartikel, die sind nur Dekoration.
- Eis und Schnee schmilzt, wenn Lichtquellen in der Nähe sind.
- Kupfer oxidiert.
- Aktiviertes Redstone-Erz wird wieder dunkel.
- Amethystknospen entstehen und wachsen.
- In kalten Biomen gefriert Wasser zu Eis.
- Bei Schneefall bleibt Schnee liegen.
- Kessel füllen sich mit Wasser, Lava oder Pulverschnee, wenn die Bedingungen stimmen.
- Bei Gewitter schlägt der Blitz ein.
- Netherportalblöcke generieren einen zombifizierten Piglin. Es sieht so aus, als wäre er durch den Nether gekommen, aber er wird tatsächlich durch ein Block-Update im Portal generiert.

Das Konzept der zufälligen Block-Updates erklärt auch ein anderes Phänomen: Wenn du im Bett schlafend die Nacht überspringst, beschleunigst du damit weder das Pflanzenwachstum noch andere Vorgänge. Das liegt daran, dass sich durch das Aufwachen nur der Sonnenstand ändert. Der Game-Tick und damit die interne Zeit laufen kontinuierlich weiter, ein Überspringen der Zeit ist nicht möglich.

LEBEN IST EINE ILLUSION – ES GIBT KEINE KREATUREN

Die Welt erscheint nicht nur durch die zufälligen Block-Updates lebendig, es gibt noch viele andere Bewegungen, die eine Illusion von Leben erzeugen. Allen voran die herumlaufenden Monster und Tiere – die besonders schnell laufen, wenn du ihnen einen Klaps gibst.

Die Lebewesen gehören zu den **Objekten** (*Entities*). Sie sind im Gegensatz zu den Blöcken, die in ihrem starren Blockgitter sitzen, frei in der Welt beweglich. Auch Fahrzeuge, fallengelassene Gegenstände, Geschosse und natürlich die Spielerfiguren gehören zu den Objekten. Jedes Objekt hat seine eigene Bewegungsrichtung und Geschwindigkeit. Daraus berechnet das Spiel mit jedem Tick für jedes Objekt die neue Position.

Objekte: Enderkristall, Gemälde, Boot, Lebewesen, fallengelassene Gegenstände, Rahmen, Pfeil in der Wand, Rüstungsständer.

Es gibt auch einige unbewegliche Objekte, z. B. Rahmen, Gemälde und Rüstungsständer. Du erkennst Objekte an ihrer Hitbox, die du mit F3 + B ein- und ausschaltest.

Lebewesen haben darüber hinaus eine sogenannte *künstliche Intelligenz* (KI), was aber nichts weiter ist als ein fest programmiertes Verhalten: Monster verfolgen dich, geschlagene Tiere fliehen, mit Futter angelockte Tiere folgen dir, Wölfe jagen Schafe, Ziegen springen, Dorfbewohner suchen ihr Bett oder ihren Arbeitsblock, Hilfsgeister suchen Gegenstände, Schafe grasen, Papageien tanzen zu Musik etc. Mit jedem Tick muss das Spiel bei jedem Objekt das aktuelle Verhalten abarbeiten, was ziemlich viel Rechenzeit kostet. Daher laufen die meisten Tiere einfach nur zufällig herum und schlafen nicht. Und damit

die Anzahl der insgesamt zu prüfenden Lebewesen nicht zu groß wird, verschwinden Monster, Wassertiere und einige rein dekorative Wildtiere wieder, sobald du dich weit genug entfernst (*Despawnen* genannt).

KI: Dorfbewohner haben einen Tagesablauf (und schlafen mit offenen Augen und Hut), Wölfe jagen Schafe.

Auch das Alter von Jungtieren und abgeschossenen Pfeilen wird mit jedem Tick geprüft. Hat ein Jungtier eine bestimmte Anzahl von Ticks erreicht, wird es durch ein erwachsenes Tier ersetzt. Abgeschossene Pfeile verschwinden nach einer gewissen Anzahl von Ticks.

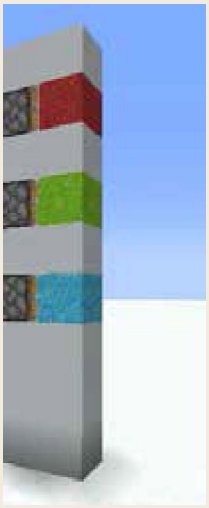
Neben den Lebewesen gibt es noch weitere bewegliche Objekte, die für eine lebendige Welt sorgen. Dazu gehören auch die fallengelassenen Gegenstände. Ein **Gegenstand** (*Item*) ist etwas, das in einem Behälter- oder Lebewesen-Inventar liegt. Lässt du einen Gegenstand fallen (*Droppen*), erzeugt das Spiel an dieser Stelle ein bewegliches Objekt, das sich langsam dreht. Hebst du das Objekt auf, wird es wieder zu einem Gegenstand.

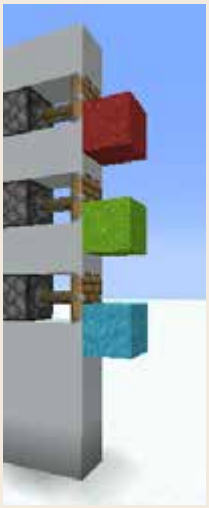
Fallengelassene Gegenstände unterliegen der Schwerkraft, können Treppen hinunterrutschen oder von Wasser weggespült werden. Auch hier berechnet das Spiel mit jedem Tick die Position und das Alter. Liegt ein fallengelassener Gegenstand 6.000 Ticks (5 Minuten) auf dem Boden, wird er gelöscht. Das Spiel muss das tun, sonst gäbe es zu viele Gegenstände auf dem Boden, deren Position berechnet werden müsste.

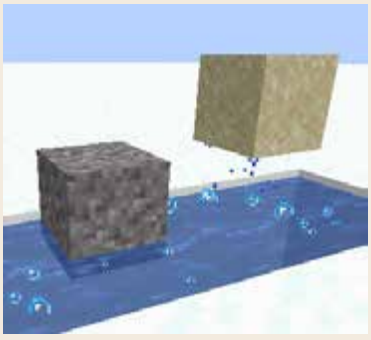
Für ein realistisches Aussehen haben Objekte einen **Objektschatten** unter sich, den du über die Minecraft-Grafikoptionen ein- und ausschalten kannst. Auch hier spart das Spiel Rechenzeit mit einem Trick: Der Schatten wird immer kreisrund dargestellt, egal wie das Objekt aussieht und woher das Licht kommt.

Objektschatten sind immer kreisrund.

Auch die simulierte *Schwerkraft* wird durch Objekte dargestellt. Das geschieht ähnlich wie bei den fallengelassenen Gegenständen durch Umwandlung: Sobald ein schwerkraftabhängiger Block wie Sand, Kies, Trockenbeton oder Amboss unter sich keinen Boden mehr hat, wird er in ein gleich aussehendes, fallendes Objekt verwandelt. Das Objekt kann sich im Gegensatz zum Block frei durch das Blockgitter nach unten bewegen (und durch Explosionen sogar zur Seite). Erst wenn das fallende Objekt sicher landet, wird es wieder zum Block.







Links und Mitte: Fallende Blöcke sind Objekte, die sich durch das Blockgitter bewegen.
Rechts: Lässt du Kies oder Sand von oben auf eine Blasensäule fallen (mit Seelensand auf dem Grund), tanzen sie auf dem Wasser.

Klar, dass diese vielen Berechnungen, die eine lebendige Welt simulieren, Rechenzeit kosten und deinen Computer belasten. Vor allem, weil sämtliche Berechnungen innerhalb eines einzigen Ticks erfolgen müssen. Um Lags zu

vermeiden, gibt es verschiedene Grafikeinstellungen, mit denen du die Grenze zwischen realistischer Darstellung und Computerentlastung nach deinen persönlichen Anforderungen selbst festlegst.

Dazu gehört auch die **Simulationsweite**. Du kennst bereits die **Sichtweite** für die Größe der geladenen Welt. Im Gegensatz dazu bestimmt die Simulationsweite, wie weit das Leben in der Welt simuliert wird. Möchtest du eine größere Sichtweite, aber dein Computer schafft das nicht, dann verringerst du die Simulationsweite. Damit kannst du Tiere, Monster, Ackerpflanzen und Setzlinge immer noch bis zum Horizont sehen, aber außerhalb der Simulationsweite verändern sie sich nicht. Du siehst mehr Landschaft, ohne deinen Computer großartig zu belasten.

Um noch mehr Rechenpower zu sparen, nutzt du die **Objektentfernung**. Damit stellst du die Sichtbarkeit von Objekten (z. B. Tiere und Monster) prozentual zur Simulationsweite ein.

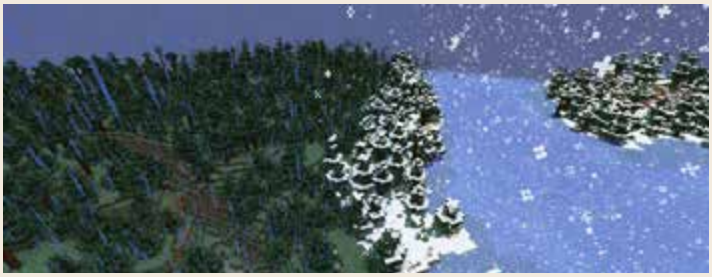
Sanfte Brisen und Schneeschauer

Ein weiteres Element für die Illusion einer belebten Natur ist das **Wetter**. Wolken ziehen in endloser Wiederholung langsam von Ost nach West über den Himmel, wodurch der Eindruck eines sachten *Windes* entsteht. Auch Banner wehen leicht in einem nicht vorhandenen Wind. Mit dem Einsatz von Shadern, die du in einem späteren Kapitel kennenlernst, bewegen sich auch Laub und Gras im Wind hin und her, aber das Originalspiel verzichtet auf diesen rechenintensiven Effekt. Zur weiteren Entlastung kannst du die Wolken in den Grafikeinstellungen einfacher machen oder ganz abschalten.

Auch der *Niederschlag* (Regen und Schnee) ist nur eine Illusion. Es regnet nicht wirklich bis zum Horizont. Stattdessen werden vor deinen Augen, und nur zehn Blöcke weit, kleine Grafikelemente eingeblendet. Der Niederschlag ist nur da, wo du hinschaust, hinter dir regnet es nie (genau über und unter dir übrigens auch nicht). Je nach Biom sieht das aus wie Wassertropfen oder Schneeflocken.

Flussbiome gehören zum »gemäßigten Klima«. Deshalb regnet es dort, auch wenn sie durch eine heiße Wüste verlaufen. Für kalte Regionen gibt es ein eigenes Fluss-Biom, den »gefrorenen Fluss«, über dem es schneit.

Für die perfekte Illusion erzeugt das Spiel zusätzlich Regengeräusche und Wasserspritz-Partikel auf dem Boden – diese allerdings auch nur ein paar Blöcke weit, das reicht fürs Auge.



Eine Biomgrenze zwischen Regen und Schnee. Der Himmel wird bei Niederschlag verdunkelt (auch in der Wüste).

Solange sich das Spiel im Regen-Zustand befindet, gibt es außerdem spezielle Block-Updates: Kessel füllen sich langsam mit Wasser, Ackerboden wird nass, Weizen wächst schneller, Feuer geht aus. In »kalten« Biomen (und ab einer gewissen Höhe) lassen Block-Updates Schnee auf dem Boden entstehen und verwandeln Wasser in Eis, während es in »heißen« Biomen nie regnet und nur der Himmel verdunkelt wird. Dadurch bekommst du den Eindruck von Nässe, Hitze und Kälte nur durch optische Effekte.

Ein Gewitter im Eismeer

Besonders eindrucksvoll sind Gewitter, die eigentlich nur eine Mischung aus sehr dunklem Himmel, lauten Geräuschen und Blitzeffekten sind. Wobei der Blitz nicht aus dem Himmel auf die Erde trifft, sondern wieder durch ein Block-Update auf der Erde erzeugt wird. Der zufällig ausgewählte Block sorgt für einen kurzen Lichtstrahl über sich, einige Feuer um sich herum und Schaden bei Kreaturen in seiner Nähe.

Eben war von »Wasserspritz-Partikeln« die Rede. **Partikel** sind kleine Grafikelemente, die auf berechneten Bahnen herumfliegen und nach kurzer Zeit verschwinden. Auch sie sorgen durch ihre Bewegung für eine lebendige Umwelt. Doch je mehr Partikel herumfliegen, desto mehr wird das Rendern belastet. Daher kannst du in den Grafikeinstellungen die Anzahl der Partikel verringern.

Einige Partikel: tropfender und liegender Honig, Endstabtropfen, Fackelrauch, Wassertropfen, Wasserspritzer auf dem Boden, Lagerfeuerrauch, wegfliegende Funken

Gratulation, du hast das 1. Experten-Level erreicht

Jetzt weißt du, was **Chunks** sind, wie du ein **Backup** machst, wie F3 deine Koordinaten anzeigt, was eine **Block-ID** ist, wie das **Rendern** funktioniert, was die **Hitboxen** bewirken, wie du den **Zuschauermodus** aktivierst, wie lange ein **Tick** dauert und was der Unterschied zwischen Blöcken, Gegenständen, Objekten und Partikeln ist.

Du hast einen ausgiebigen Blick hinter die Kulissen geworfen und kennst die Tricks, mit denen das Spiel arbeitet:

- **Nebel** verbirgt die Tatsache, dass die geladene Welt direkt hinter dem Horizont endet,
- durch das **Rendern** hast du überall, wo du hinschaust, ein schnell ablaufendes Daumenkino aus Bildern vor dir, obwohl die Welt in Wirklichkeit nur ein farbloses Blockgitter ist,
- eine **interne Uhr** tickt mit rasender Geschwindigkeit, berechnet ständig neue Positionen von beweglichen Objekten und erzeugt um dich herum ein Gewitter von zufälligen Block-Updates, die mit ihren seltenen Treffern für langsame Veränderungen sorgen,
- und schließlich wirkt alles zusammen und gaukelt dir durch zahlreiche Bewegungen von Kreaturen, Partikeln und Wettersimulationen eine riesige, **lebendige Welt** vor.

Und obwohl du eigentlich nur unrealistische, bunte Blöcke und seltsame Kreaturen siehst, bist du nach kurzer Zeit völlig versunken und glaubst, in einer realen Welt zu sein. It's magic.

In den nächsten Kapiteln erfährst du, wie du diese magische Welt für dein Survivalspiel noch weiter verschönerst. Als Erstes verpasst du dir und der Welt ein neues Aussehen.