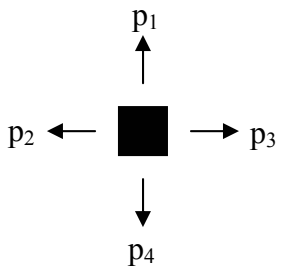
Zitterfahrten, Kristalle, Chaos, Ordnung?

Werden Teilchen in einem Molekül unter dem Mikroskop betrachtet, so fällt eine äusserst unregelmässige Zitterbewegung auf, die Brown'sche Bewegung. So eine Zitterbewegung wollen wir auf dem Computer simulieren.
Wir nehmen an, dass ein Molekül am Anfang still steht und sich bei jedem Schritt entweder einen Schritt nach links, rechts, oben oder unten bewegt. In welche Richtung sich das Molekül bewegt, wird von den 4 Wahrscheinlichkeiten p_1, p_2, p_3 und p_4 bestimmt.



Die Wahrscheinlichkeiten ergeben zusammenaddiert immer 1. Im Normalfall sind die Wahrscheinlichkeiten alle gleich, also $p_1 = p_2 = p_3 = p_4 = 0.25$. Bei jedem Schritt wird neu gewürfelt und eine neue Verschiebungsrichtung bestimmt.

Aufgabe 1

Benutze das Gerüst zu Zitterfahrten. Es sieht ähnlich aus wie dein Busplan.java.
Realisiere nun eine solche Zitterfahrt, indem du die oben beschriebene Iterationsvorschrift programmierst.
Hinweis zum „Würfeln der Richtung": Betrachte die Wahrscheinlichkeiten als Abschnitt einer 1 Einheit langen Strecke. Wähle eine Zufallszahl zwischen 0 und 1, die du einem Streckenabschnitt zuordnest. Eine Zufallszahl kannst du mit der Methode `zufallsZahl()` erzeugen.

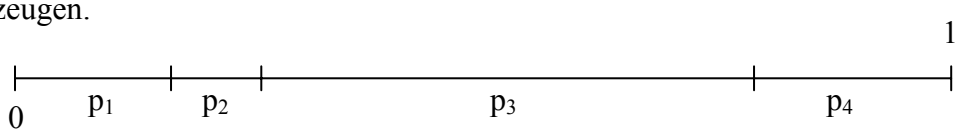


Abbildung 1: Zitterfahrt mit 100'000 Schritten

Aufgabe 2

Mit Hilfe der Zitterfahrten kann man sehr schöne geometrische Figuren erzeugen, die in der Realität beim Wachstum von Kristallen vorkommen. Bei Kristallen wird ein kleiner Impfkristall benötigt, an dem der Kristall wächst. Wir benötigen dazu einfach einen schwarzgefärbten Bildpunkt in der Mitte des Bildschirmes.
Als nächstes benötigen wir ein Feld des Bildschirmes, das wir als Startpunkt für eine unsichtbare Zitterfahrt benötigen. Wir berechnen uns also eine zufällige i- bzw. j-Koordinate. Wir haben wieder die Methode `zufallsZahl()` zur Verfügung. Da diese Methode Werte zwischen 0 und 1 zurückgibt, müssen wir sie auf Fensterbreite/-höhe hochskalieren.
Wir starten nun eine unsichtbare Zitterfahrt, die solange dauert, bis das Nachbarfeld eines Punktes (i_{paint},j_{paint}) schwarz gefärbt ist. Dazu brauchen wir die Methode `boolean isPixel(int i, int j)`, die uns angibt, ob der Punkt (i,j) schwarz gefärbt ist. Ist ein Nachbarpunkt von (i_{paint},j_{paint}) schwarz gefärbt, so färben wir (i_{paint},j_{paint}) auch schwarz. Nun ist ein weiterer Punkt zum Kristall hinzugekommen.
Danach wählen wir ein neues Feld (wieder zufällig) auf dem Bildschirm und wir wiederholen alles – eine neue Iteration beginnt.
Hinweis: Um einen schönen Kristall zu bekommen, sind etwa 10’000 bis 100'000 Iterationen notwendig.

Aufgabe 3

Anstatt die 4 Wahrscheinlichkeiten p_1, p_2, p_3 und p_4 alle gleich zu wählen, kann man sie auch variieren. Oder man kann statt dem Punkt als Impfkristall eine horizontale Linie, ein Maschengitter, etc. wählen. So könnte man z.B. die Tropfenbildung von Nebel an einem Drahtgitter simulieren. Spiele mit den Formen und Werten herum!