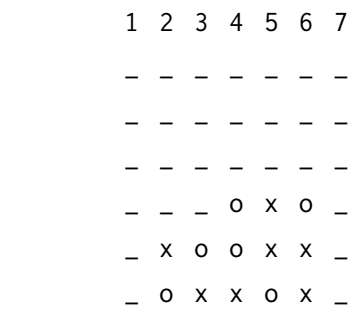
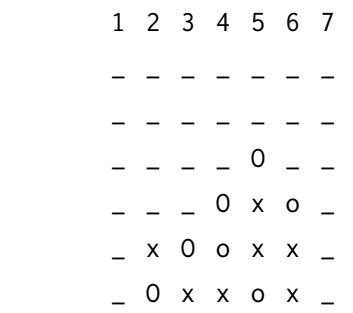
v13r g3w1nn7

Gamedesign-Portfolio: Vier gewinnt für Sysadmins

Patrick Bucher

29. Mai 2020





Abstract

«Vier gewinnt» ist ein weit verbreitetes und bekanntes Strategiespiel für zwei Personen, bei der es darum geht, vier Spielsteine auf einem 7x6-Raster in eine horizontale, vertikele oder diagonale Reihe zu bringen. Zu Zeiten der Corona-Krise lässt sich das Spiel schlecht in physischer Gegenwart an einem Tisch spielen. Ausserdem schaffen die altbekannten Spielregeln und -mechaniken kaum noch einen Anreiz für das Spiel. In der vorliegenden Arbeit soll das Spiel «Vier gewinnt» mit neuen Mechaniken interessanter und besonders auf die Personengruppe der Systemadministratoren angepasst werden, die das Spiel lieber auf der Kommandozeile als in der physichen Version spielen.

Inhaltsverzeichnis

1 Erste Iteration: Zieldefinition und Konzept

Systemadministratoren (kurz: Sysadmins) haben es nicht einfach zu Zeiten der Corona-Krise: Ins Home-Office verbannt sind sie teilweise ihrer letzten sozialen Kontakte beraubt, zumal *Star Wars*-Conventions und andere Veranstaltungen für Nerds alle abgesagt worden sind – und dies auf eine lange Zeit hinaus. Als kleinen Trost sollen sie ein kleines Computerspiel bekommen, das im Rahmen dieses Projekts konzipiert und entwickelt wird. Dabei handelt es sich um eine erweiterte Version des Spiels «Vier gewinnt» (siehe Abbildung 1, Seite 4).

Sysadmins haben spezielle Bedürfnisse.[1] Das herkömmliche «Vier gewinnt» ist ihnen zu langweilig, zumal sie sämtliche Speilverläufe schon längst nachtsüber auf einem ihrer High-End-Server durchgerechnet haben. Zudem ist ihnen nach Spielende das lästige Sortieren der Steine nach Farben, das beim «Vier gewinnt» im Meatspace nötig ist, ein Graus, zumal Sysadmins solche mechanischen Abläufe lieber automatisieren. Ausserdem lassen sich die Spielverläufe beim physischen Spiel nur sehr umständlich und schlecht maschinenlesbar festhalten, etwa mittels Fotokamera.

1.1 Vorteile der Kommandozeilenversion

«Vier gewinnt» wäre viel besser, wenn man es auf der Kommandozeile spielen könnte! Dies hätte folgende Vorteile gegenüber der herkömmlichen Meatspace-Version:

1. Das Spiel könnte während der Arbeitszeit gespielt werden, ohne dass der gelegentlich vorbei- und in den Bildschirm schauende Vorgesetzte diese Strategie der Arbeitsvermeidung als solche erkennen könnte. Schliesslich handelt es sich ja bloss um irgendwelche kryptisch anmutende Zeichen, die da auf dem Bildschirm zu sehen sind. Bestimmt hat der Sysadmin da einen Belegungsplan für Pins oder dergleichen auf dem Bildschirm, und selbstverständlich handelt es sich dabei um geistige Schwerstarbeit, die dem Arbeitgeber zugute kommt – das verrät schon des Sysadmins angestrengter Gesichtsausdruck!

2. Die Aufforderung der Geschäftsleitung, mehr zu kommunizieren und sich Probleme gemeinsam anzuschauen, könnte für weitere Arbeitsvermeidung ausgenutzt werden, indem sich zwei Sysadmins am gleichen Computer treffen, um dort «Vier

[1] Als Sysadmins werden im Rahmen der vorliegenden Arbeit nur die Mitglieder der höchsten Kaste bezeichnet – die Unix-Systemadministratoren. Auf die Bedürfnisse der Angehörigen tieferer Kasten soll hier nicht eingegangen werden.

Abbildung 1: Das klassische «Vier gewinnt» (Meatspace-Version). Die grellen Farben der VGA-Palette überfordern das auf die monochrome Bildschirmdarstellung optimierte Auge eines Sysadmins. Das analoge Einwerfen der Steine in die Schächte weckt traumatische Erinnerungen an die Handhabung einer Computermaus.

gewinnt» (bzw. `v13r g3w1nnt!!!11`) zu spielen. Die kryptisch anmutende Pin-Belegung, die da für den Vorgesetzten auf dem Bildschirm zu sehen ist, schreit ja geradezu nach einer *Pair Debugging Session*! So kann die Arbeitszeit weiter mit «Vier gewinnt» vertrödelt werden, und die sozialen Kontakte der Firma werden dadurch noch weiter gestärkt – ganz im Sinne der Geschäftsleitung!

3. Da solche sozialen Interaktionen im Meatspace derzeit wegen *Social Distancing* kaum denkbar sind, müsste das Spiel auch remote von zwei Sysadmins spielbar sein, ohne dass die Nerds der Softwareentwicklung das Spiel aufwändig umprogrammieren müssen. So könnten die Sysadmins das Spiel von zu Hause aus über eine gemeinsame Session mit `tmux` oder GNU `screen` gegeneinander spielen. Dies dürfte auch die VPN-Verbindung ins Büro nicht weiter beeinträchtigen, zumal diese schon durch BitTorrent-Downloads voll ausgelastet ist. Der Multiplayer-Modus wäre somit auch über eine serielle Verbindung möglich, die der erfahrene Sysadmin notfalls auch aus im Büro gestohlenen Büroklammern und aus abgerissenen Streifen seines Aluhuts herstellen könnte. Sollte der Bildschirm des Sysadmins unerwartet in die Brüche gehen, wäre das Spiel immer noch über einen Fernschreiber spielbar, der sich bei jedem guten Systemadmin im Keller finden lässt.[2]

4. Die Spielverläufe können einfach in Textdateien[3] festgehalten werden. Das erleichtert die Diskussion über Spielstrategien auf dem Usenet und in IRC.

Das klassische «Vier gewinnt» müsste hierzu natürlich noch etwas interessanter ausgestaltet werden, sodass die Motivation der Sysadmins zumindest das Warten auf den nächsten Download auf den derzeit chronisch überlasteten Steam-Servern überdauert. Diese möglichen Verbesserungen sollen im Rahmen dieser Arbeit vorgeschlagen, umgesetzt und evaluiert werden.

1.2 Zwei Tastaturen, eine Kommandozeile

Um das Treffen zweier Sysadmins auf einem gemeinsamen Terminal, das für die Evaluation des Prototyps nötig ist, zu bewerkstelligen, muss ein gemeinsam zugänglicher Server zuerst entsprechend konfiguriert werden.[4]

[2] In der Regel hinter der Kiste mit den *Star Trek*-Fanartikeln.

[3] d.h. MIME-Type `text/html;charset=utf-8`, selbstverständlich ohne *byte order mark* (BOM)

[4] Server können von HSLU-Studierenden kostenlos und unkompliziert im *Enterprise Lab* bezogen werden. Diese dürften knapp genügend Leistung aufbringen, um eine Anbindung mit einer Baudrate von 9600

Sollte wider aller Erwartung kein Solaris, sondern ein Debian-basiertes Betriebssystem auf dem Server installiert sein, lässt sich die gemeinsame Spielumgebung mit den folgenden Schritten einrichten.

Zunächst muss `tmux` installiert werden:

```
# apt-get install tmux
```

Als nächstes ist eine gemeinsame Benutzergruppe und je ein Benutzer für die beiden Spieler (`geek` und `poke`) zu erstellen. Auch müssen den neu erstellen Benutzern gleich Passwörter hinterlegt werden:

```
# groupadd sysadmins
# useradd -G sysadmins -m geek
# useradd -G sysadmins -m poke
# passwd geek
# passwd poke
```

Das `$HOME`-Verzeichnis, das über das `-m`-Flag erstellt wird, ist wichtig, da die beiden Spieler ihre Spielverläufe ja schliesslich in Textdateien festhalten und kommentieren wollen – letzteres vorzugsweise mit `ed(1)`, dem Standard-Texteditor von Unix.

Nun kann sich der erste Benutzer (`geek`) auf dem Server einloggen und eine neue `tmux`-Session namens `game` eröffnen:

```
[geek]$ tmux -S /tmp/shared new -s game
```

Die Session muss so berechtigt werden, dass alle Benutzer der Gruppe `sysadmins` darauf zugreifen können:

```
[geek]$ chgrp sysadmins /tmp/shared
```

Anschliessend kann sich der zweite Benutzer (`poke`) einloggen und an der bestehenden `tmux`-Session teilnehmen:

```
[poke]$ tmux -S /tmp/shared attach -t game
```

Jetzt benötigen die beiden Spieler bloss noch ein Spiel, und der Spass kann auf einer tiefen Bandbreite beginnen!

über längere Zeit zu gewährleisten. Mit neueren Versionen von Solaris dürfte sich auch `tmux` einfach installieren lassen (siehe `https://www.opencsw.org/packages/tmux/`).

1.3 Prototyp für «Vier gewinnt»

In dieser ersten Iteration soll nur das herkömmliche «Vier gewinnt»-Spiel zum Einsatz kommen. Dieses wird zunächst in einer Kommandozeilenversion entwickelt.

Als Umsetzungsplattform für den Prototyp kommt Python 3 zum Einsatz. Mit dieser Programmiersprache hat der Autor dieser Arbeit bereits einiges an positiver Erfahrung sammeln können. Ausserdem verfügt Python mit NumPy[5] über eine mächtige Library für (mehrdimensionale) Arrays, mit der sich viele Listenoperationen mit wenig Code und effizient umsetzen lassen.[6]

Die Spiellogik wird mithilfe von pytest[7] automatisiert getestet. Auch wenn es sich nur um einen Prototyp handelt, ist die Korrektheit der Spiellogik ein wichtiger Usability-Faktor, zumal nicht erkannte Gewinnsituationen für die Beteiligten sehr frustrierend sind, was einen negativen Einfluss auf die Testläufe hätte.

Der Prototyp ist einfach zu bedienen und schliesst Fehleingaben so gut wie möglich aus. Das Folgende Kommandozeilenfenster zeigt den Prototyp während eines Spiels:

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ x _ _ o _ _
_ o x o o o x
_ x o x x x o

Player 1 [x] (1 to 7): ¶
```

Das Spielfeld wird mit einem 7x6-Raster dargestellt. Der Underscore (_) markiert dabei leere Felder. Spieler 1 spielt mit dem Spielstein x, und Spieler 2 mit dem Spielstein o. Die Titelzeile markiert die einzelnen Spalten (oder «Schächte») mit einer Nummer, sodass sich der Spieler beim Auswählen des Schachts einfacher orientieren kann.

Das Paragraph-Zeichen (¶) markiert die Eingabeaufforderung. Der Spieler soll die Nummer des Schachts auswählen, in den sein Spielstein fallen soll. Falscheingaben (nicht numerische, nicht im Bereich 1-7, einen vollen Schacht bezeichnend usw.) werden nicht ak-

[5] https://numpy.org/

[6] Für die richtige Version des Spiels würde eine Programmiersprache verwendet werden, die eine höhere Laufzeitperformance bei geringerem Speicherverbrauch ermöglicht, wie z.B. ANSI C 89, Assembler oder Fortran 66, damit der Sysadmin sich nicht unnötig Rechenpower für das Bitcoin-Mining verliert.

[7] https://docs.pytest.org/en/latest/

zeptiert, sondern mit einer Fehlermeldung und einer erneuten Eingabeaufforderung quittiert:

```
1 2 3 4 5 6 7
_ _ _ _ x _ _
_ _ _ _ o _ _
_ _ _ _ x _ _
_ x _ _ o _ _
_ o x o o o x
_ x o x x x o

Player 2 [o] (1 to 7): 5
You must pick a free slot between 1 and 7!
```

Sind alle Spielfelder voll, ohne dass dabei ein Spieler die Siegbedingung (vier Spielsteine in einer horizontalen, vertikalen oder diagonalen Reihe) erfüllt, wird das Spiel als unentschieden gewertet. Erfüllt ein Spieler die Siegbedingung, ist das Spiel zu Ende und der Sieger wird als solcher ausgewiesen:

```
1 2 3 4 5 6 7
_ _ _ _ x _ _
_ _ _ _ o _ _
_ _ _ _ X _ _
_ x _ X o o _
_ o X o o o x
_ X o x x x o

Player 1 has won the game.
```

Hierbei wird zur leichteren Erkennung die Siegreihe hervorgehoben, indem der Spielstein durch einen entsprechenden Grossbuchstaben ersetzt wird (`x` wird hier durch `X` ersetzt). Dies soll Diskussionen zwischen aufbegrachten Sysadmins verhindern, die ansonsten nur in die Richtung gehen würde, dass diese Software halt falsch programmiert worden sei – und die Erkennung der Siegbedingung nicht ausreichend getestet.

Das Spiel konnte in dieser Grundversion erfolgreich getestet werden. Es ist somit bereit für die Weiterentwicklung, die im nächsten Kapitel besprochen wird.

2 Zweite Iteration: Anwendung von Game-Design-Theorie

2.1 Theorie

TODO: welche Theorie wird verwendet?

2.1.1 Entscheidungen

Verschiedene Spiele lassen verschiedene Arten von Entscheidungen zu. Bei einfachen Spielen hat man es v.a. mit uninteressanten Entscheidungen zu tun. Diese lassen sich in die folgenden Kategorien aufteilen:

Keine Entscheidungen Das Würfeln bei einem Leiterlispiel ist keine Entscheidung, da hier der Zufall regiert. Bei jeder Runde muss gewürfelt werden, der Spieler kann darüber nicht entscheiden.

Bedeutungslose Entscheidungen Das Tippen auf Kopf oder Zahl beim Münzwurf ist eine bedeutungslose Entscheidung. Es muss eine Entscheidung getroffen werden, die das Spielergebnis aber nicht beeinflusst, bzw. vom Glück abhängig ist.

Offensichtliche Entscheidungen Das setzen eines Steines in «Vier gewinnt» ist eine offensichtliche Entscheidung. Aus der Spielmechanik geht etwa klar hervor, welche Eröffnungszüge sinnvoll sind (in der Mitte spielen), oder welche Züge den Spieler dem Sieg näherbringen (Einreihung von Steinen).

Blinde Entscheidungen Das Wählen einer Koordinate beim «Schiffe versenken» läuft zumeist blind ab, da man das Spielfeld des Gegners nicht kennt. Im Verlauf des Spiels erhält man jedoch immer ein besseres Bild vom Spielfeld des Gegners, sodass in manchen Spielsituationen das Versenken von grossen Schiffen zu einer offensichtlichen Entscheidung werden kann.

Beim Spiel «Vier gewinnt» hat man es mit offensichtlichen Entscheidungen zu tun. Umd as Spiel interessanter zu machen, muss es interessante Entscheidungen zulassen.

Interessante Entscheidungen unterscheiden sich von uninteressanten Enscheidungen u.a. dadurch, dass mit ihnen ein gewisses Risiko einhergeht, das sich nicht oder nur begrenzt abschätzen lässt. Oft wirken sich risikoreiche Entscheidungen nicht schon unmittelbar nach der Entscheidung, sondern erst später auf den Spielverlauf aus.

Bei «Vier gewinnt» steht dem Spieler nur eine Art von Auswahl zur Verfügung: Die Wahl des Schachts, in der ein Stein eingeworfen werden soll. Das Spiel kann dann interessanter werden, wenn andere Arten von Entscheidungen angeboten werden. Je nach Spielsituation wäre dann eine andere Art der Entscheidung angebracht.

2.1.2 Randomness

TODO: Theorie de Randomness zusammenfassen

2.2 Verbesserung des Prototyps

2.2.1 Rotation als interessante Entscheidung

2.2.2 Neutrale Steine als Zufallselement

3 Dritte Iteration: Abschliessender Nutzertest und Reflexion

3.1 Durchführung der Tests

Mit n Leuten m Runden gespielt

3.2 Rückmeldungen der Spieler

3.3 Fazit

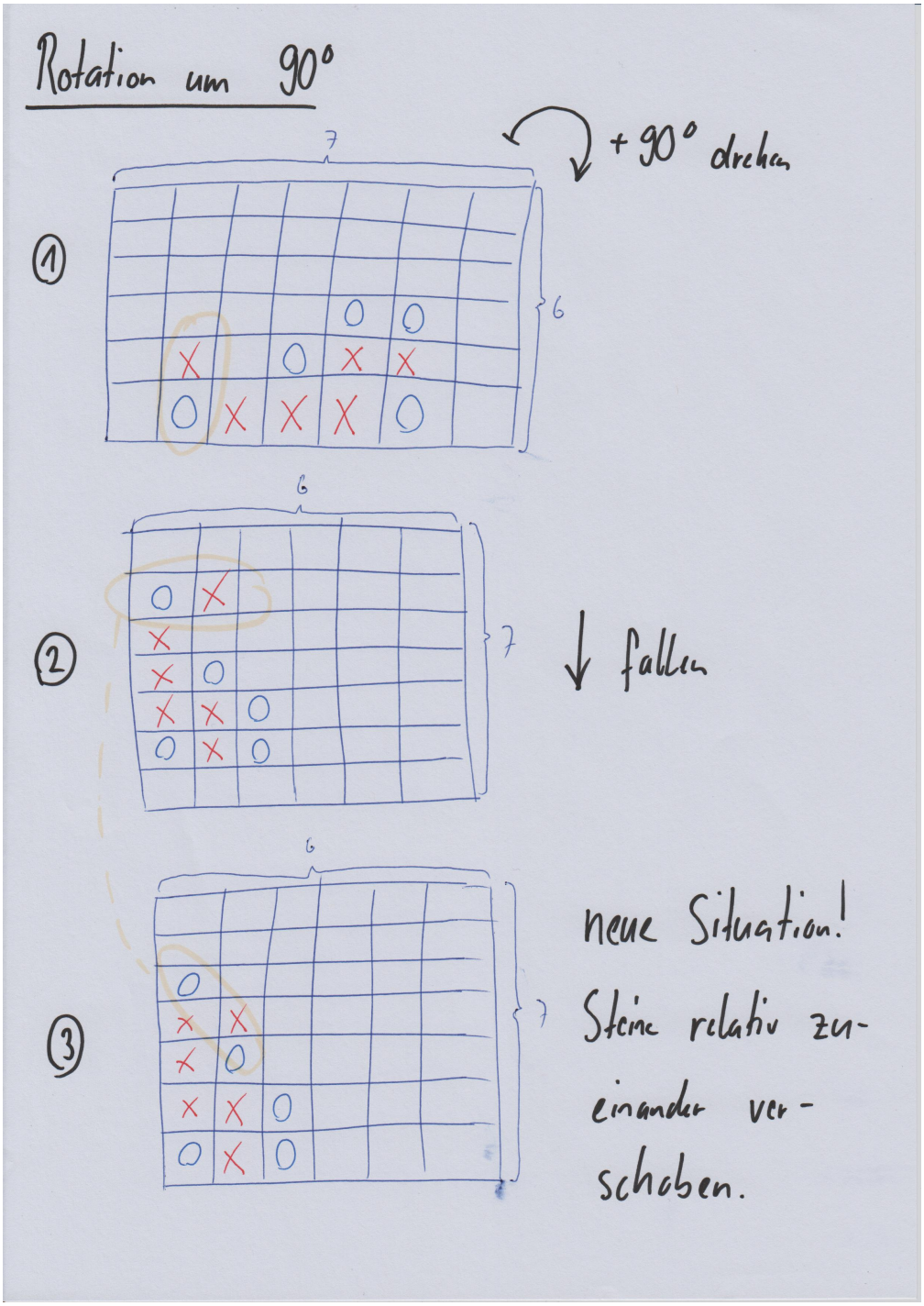


Abbildung 2: Die Rotation um 90° als zusätzliche Spielmechanik. Das Spielgitter wird zunächst gedreht, anschliessend fallen die Steine nach unten. Dadurch entsteht eine neue Spielsituation, zumal sich manche Steine relativ zueinander bewegen.

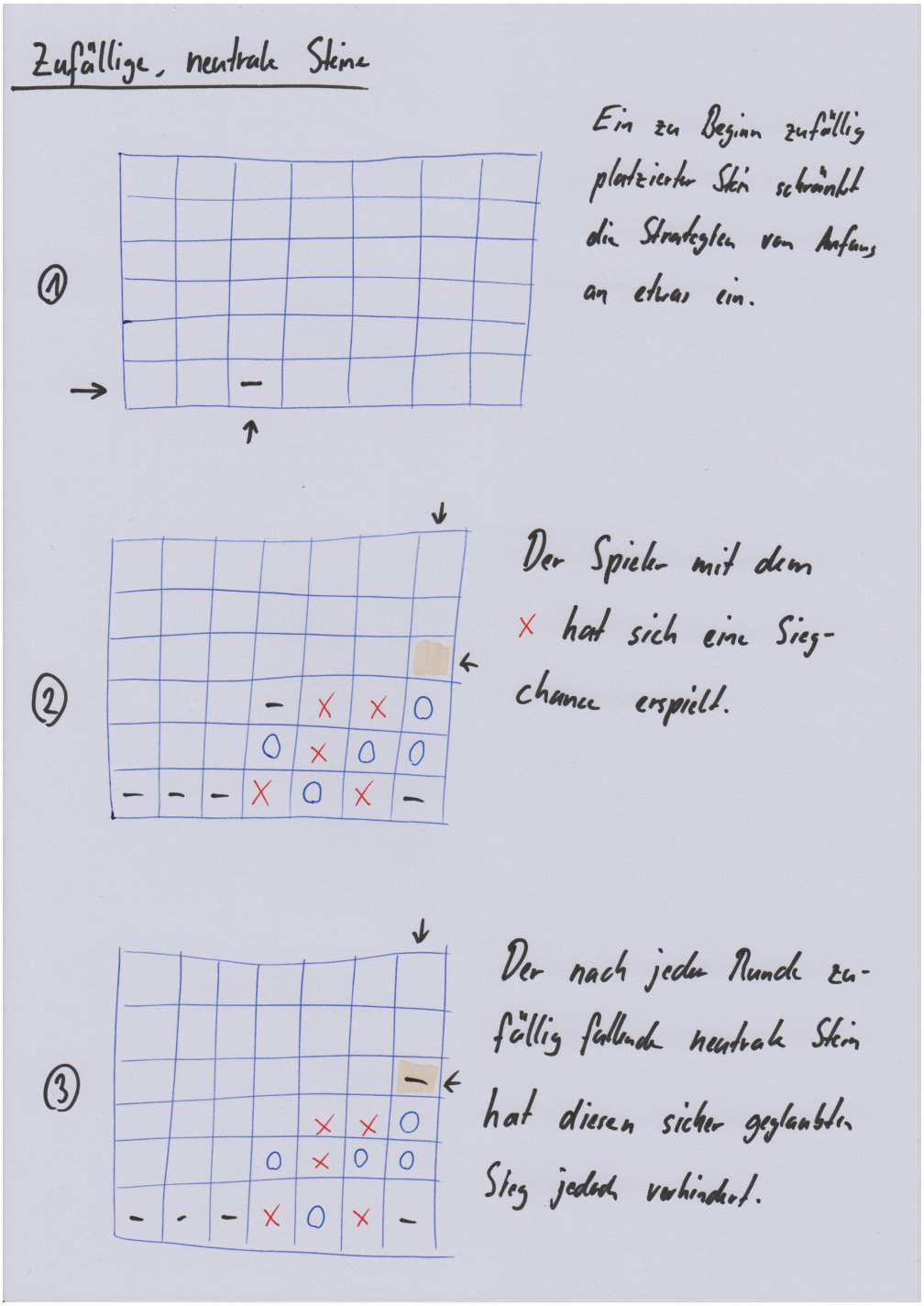


Abbildung 3: Zufällig fallende, neutrale Steine sollen dem Spiel ein Zufallselement hinzufügen. Dadurch werden etablierte Strategien der Spieler möglicherweise durchkreuzt; sie müssen improvisieren.

Abbildungsverzeichnis