

# Agentenmodell: Neue Angebot Verbreitung

---

 [chatgpt.com/c/089ba15a-3e12-47cd-a7ac-5c88bb0031fb](https://chatgpt.com/c/089ba15a-3e12-47cd-a7ac-5c88bb0031fb)



ChatGPT

✓

Schreibe eine einfache Agenten-basierte Simulation in NetLogo, mit der die Verbreitung der Nutzung eines neuen Angebots simuliert werden kann.



NetLogo ist eine leistungsstarke Plattform zur Erstellung von Agenten-basierten Modellen und Simulationen. Hier ist ein einfaches Modell, das die Verbreitung der Nutzung eines neuen Angebots simuliert.

In diesem Modell gibt es zwei Arten von Agenten: "Nutzer" und "Nicht-Nutzer". Jeder Agent hat eine gewisse Wahrscheinlichkeit, ein Nutzer zu werden, basierend auf der Anzahl der Nachbarn, die bereits Nutzer sind.

## NetLogo Code

---

netlogo

; Erstellen Sie eine neue NetLogo-Simulation und kopieren Sie den folgenden Code in den Code-Bereich.

```
globals [initial-users adoption-prob]
```

```
turtles-own [user?]
```

```
to setup
  clear-all
  setup-globals
  setup-turtles
  reset-ticks
end
```

```
to setup-globals
  set initial-users 10           ; Anzahl der Anfangsnutzer
  set adoption-prob 0.3         ; Wahrscheinlichkeit, dass ein Nicht-
  Nutzer ein Nutzer wird
end
```

```
to setup-turtles
  create-turtles 100 [           ; Erstelle 100 Schildkröten
    setxy random-xcor random-ycor
    set user? false             ; Standardmäßig sind alle Schildkröten
    keine Nutzer
    set color blue              ; Nicht-Nutzer sind blau
  ]
  ask n-of initial-users turtles [ ; Wählen Sie zufällig eine Anzahl von
  Schildkröten als anfängliche Nutzer
    become-user
  ]
end
```

```
to go
  ask turtles [ evaluate-adoption ]
  tick
end
```

```
to evaluate-adoption
  if not user? [                 ; Wenn die Schildkröte kein Nutzer ist
    let user-neighbors count neighbors with [user?]
    if (user-neighbors > 0) and (random-float 1 < adoption-prob) [
      become-user
    ]
  ]
end
```

```
to become-user
```

```

    set user? true
    set color green          ; Nutzer sind grün
end

; Setup und Go Buttons hinzufügen:
; 1. Erstellen Sie einen "Setup" Button und verknüpfen Sie ihn mit dem
"setup" Befehl.
; 2. Erstellen Sie einen "Go" Button und verknüpfen Sie ihn mit dem "go"
Befehl, aktivieren Sie "Forever".

; Schieberegler hinzufügen:
; 1. Erstellen Sie einen Schieberegler namens "initial-users" mit einem
Bereich von 1 bis 100 und einem Anfangswert von 10.
; 2. Erstellen Sie einen Schieberegler namens "adoption-prob" mit einem
Bereich von 0.0 bis 1.0 und einem Anfangswert von 0.3.

```

## Erklärung des Modells

---

1. **Globals:** Es werden globale Variablen definiert, die die anfängliche Anzahl der Nutzer und die Adoptionswahrscheinlichkeit speichern.
2. **Turtles-own:** Jede Schildkröte hat eine Eigenschaft `user?`, die speichert, ob die Schildkröte ein Nutzer ist oder nicht.
3. **Setup:**
  - **setup-globals:** Initialisiert die globalen Variablen.
  - **setup-turtles:** Erstellt eine bestimmte Anzahl von Schildkröten, platziert sie zufällig im Raum und setzt ihre anfänglichen Eigenschaften.
4. **go:** In jeder Taktphase überprüfen die Schildkröten, ob sie Nutzer werden sollten, basierend auf der Anzahl ihrer Nachbarn, die Nutzer sind.
5. **evaluate-adoption:** Wenn eine Schildkröte kein Nutzer ist, überprüft sie die Anzahl der Nachbarn, die Nutzer sind. Wenn diese Anzahl größer als Null ist und die Zufallszahl unter der Adoptionswahrscheinlichkeit liegt, wird sie ein Nutzer.
6. **become-user:** Setzt die Schildkröte als Nutzer und ändert ihre Farbe.

Durch Anpassung der Parameter `initial-users` und `adoption-prob` können Sie das Verhalten der Simulation verändern und beobachten, wie sich die Verbreitung des neuen Angebots entwickelt.

You can't use `USER?` in patch context, because `USER?` is turtle-only.



Du hast recht. Die Fehlermeldung weist darauf hin, dass der Befehl `user?` nur im Kontext von Schildkröten verwendet werden kann. Der Fehler tritt auf, weil die Nachbarschaftsüberprüfung (`neighbors`) auf Patches und nicht auf Schildkröten anwendbar ist.

Hier ist der korrigierte Code für die Simulation:

netlogo

```
globals [initial-users adoption-prob]
```

```
turtles-own [user?]
```

```
to setup
  clear-all
  setup-globals
  setup-turtles
  reset-ticks
end
```

```
to setup-globals
  set initial-users 10          ; Anzahl der Anfangsnutzer
  set adoption-prob 0.3        ; Wahrscheinlichkeit, dass ein Nicht-
  Nutzer ein Nutzer wird
end
```

```
to setup-turtles
  create-turtles 100 [          ; Erstelle 100 Schildkröten
    setxy random-xcor random-ycor
    set user? false             ; Standardmäßig sind alle Schildkröten
    keine Nutzer
    set color blue              ; Nicht-Nutzer sind blau
  ]
  ask n-of initial-users turtles [ ; Wählen Sie zufällig eine Anzahl von
  Schildkröten als anfängliche Nutzer
    become-user
  ]
end
```

```
to go
  ask turtles [ evaluate-adoption ]
  tick
end
```

```
to evaluate-adoption
  if not user? [                ; Wenn die Schildkröte kein Nutzer ist
    let user-neighbors count turtles-on neighbors4 with [user?]
    if (user-neighbors > 0) and (random-float 1 < adoption-prob) [
      become-user
    ]
  ]
end
```

```
to become-user
  set user? true
  set color green              ; Nutzer sind grün
end
```

```
; Setup und Go Buttons hinzufügen:  
; 1. Erstellen Sie einen "Setup" Button und verknüpfen Sie ihn mit dem  
"setup" Befehl.  
; 2. Erstellen Sie einen "Go" Button und verknüpfen Sie ihn mit dem "go"  
Befehl, aktivieren Sie "Forever".  
  
; Schieberegler hinzufügen:  
; 1. Erstellen Sie einen Schieberegler namens "initial-users" mit einem  
Bereich von 1 bis 100 und einem Anfangswert von 10.  
; 2. Erstellen Sie einen Schieberegler namens "adoption-prob" mit einem  
Bereich von 0.0 bis 1.0 und einem Anfangswert von 0.3.
```

## Erklärung der Änderungen

---

1. **turtles-on neighbors4 with [user?]**: Dies stellt sicher, dass die Überprüfung nur auf den Schildkröten und nicht auf den Patches stattfindet. **neighbors4** wählt die vier direkt benachbarten Patches (oben, unten, links, rechts), und **turtles-on** wendet den Filter auf die Schildkröten in diesen Patches an.

Mit diesen Anpassungen sollte die Simulation wie gewünscht funktionieren.

2/2

You can't use USER? in patch context, because USER? is turtle-only.



Du hast recht. Die Verwendung von **neighbors4** bezieht sich auf Patches und nicht direkt auf Schildkröten. Um dies zu korrigieren, können wir **in-radius** verwenden, um die benachbarten Schildkröten zu finden. Hier ist der angepasste NetLogo-Code:

```
netlogo
```

```
globals [initial-users adoption-prob]
```

```
turtles-own [user?]
```

```
to setup
  clear-all
  setup-globals
  setup-turtles
  reset-ticks
end
```

```
to setup-globals
  set initial-users 10          ; Anzahl der Anfangsnutzer
  set adoption-prob 0.3        ; Wahrscheinlichkeit, dass ein Nicht-
  Nutzer ein Nutzer wird
end
```

```
to setup-turtles
  create-turtles 100 [          ; Erstelle 100 Schildkröten
    setxy random-xcor random-ycor
    set user? false             ; Standardmäßig sind alle Schildkröten
    keine Nutzer
    set color blue              ; Nicht-Nutzer sind blau
  ]
  ask n-of initial-users turtles [ ; Wählen Sie zufällig eine Anzahl von
  Schildkröten als anfängliche Nutzer
    become-user
  ]
end
```

```
to go
  ask turtles [ evaluate-adoption ]
  tick
end
```

```
to evaluate-adoption
  if not user? [                ; Wenn die Schildkröte kein Nutzer ist
    let user-neighbors count turtles in-radius 1 with [user?]
    if (user-neighbors > 0) and (random-float 1 < adoption-prob) [
      become-user
    ]
  ]
end
```

```
to become-user
  set user? true
  set color green              ; Nutzer sind grün
end
```

```
; Setup und Go Buttons hinzufügen:  
; 1. Erstellen Sie einen "Setup" Button und verknüpfen Sie ihn mit dem  
"setup" Befehl.  
; 2. Erstellen Sie einen "Go" Button und verknüpfen Sie ihn mit dem "go"  
Befehl, aktivieren Sie "Forever".  
  
; Schieberegler hinzufügen:  
; 1. Erstellen Sie einen Schieberegler namens "initial-users" mit einem  
Bereich von 1 bis 100 und einem Anfangswert von 10.  
; 2. Erstellen Sie einen Schieberegler namens "adoption-prob" mit einem  
Bereich von 0.0 bis 1.0 und einem Anfangswert von 0.3.
```

## Erklärung der Änderungen

---

1. **turtles in-radius 1 with [user?]**: Dies wählt die benachbarten Schildkröten innerhalb eines Radius von 1 Patch aus und überprüft, ob sie Nutzer sind.

Mit dieser Änderung sollten die Schildkröten nun korrekt die Nachbarn überprüfen und die Simulation sollte ohne Fehler funktionieren.

Das Programm lautet:



Entschuldigung für die Verwirrung. Hier ist der korrigierte Code gemäß Ihrer Anforderung:



netlogo

```
globals [initial-users adoption-prob]
```

```
turtles-own [user?]
```

```
to setup
  clear-all
  setup-globals
  setup-turtles
  reset-ticks
end
```

```
to setup-globals
  set initial-users 10           ; Anzahl der Anfangsnutzer
  set adoption-prob 0.3         ; Wahrscheinlichkeit, dass ein Nicht-
  Nutzer ein Nutzer wird
end
```

```
to setup-turtles
  create-turtles 100 [           ; Erstelle 100 Schildkröten
    setxy random-xcor random-ycor
    set user? false             ; Standardmäßig sind alle Schildkröten
    keine Nutzer
    set color blue              ; Nicht-Nutzer sind blau
  ]
  ask n-of initial-users turtles [ ; Wählen Sie zufällig eine Anzahl von
  Schildkröten als anfängliche Nutzer
    become-user
  ]
end
```

```
to go
  ask turtles [ evaluate-adoption ]
  tick
end
```

```
to evaluate-adoption
  if not user? [                 ; Wenn die Schildkröte kein Nutzer ist
    let user-neighbors count other turtles-here with [user?]
    if (user-neighbors > 0) and (random-float 1 < adoption-prob) [
      become-user
    ]
  ]
end
```

```
to become-user
  set user? true
  set color green                ; Nutzer sind grün
end
```

```

; Setup und Go Buttons hinzufügen:
; 1. Erstellen Sie einen "Setup" Button und verknüpfen Sie ihn mit dem
"setup" Befehl.
; 2. Erstellen Sie einen "Go" Button und verknüpfen Sie ihn mit dem "go"
Befehl, aktivieren Sie "Forever".

; Schieberegler hinzufügen:
; 1. Erstellen Sie einen Schieberegler namens "initial-users" mit einem
Bereich von 1 bis 100 und einem Anfangswert von 10.
; 2. Erstellen Sie einen Schieberegler namens "adoption-prob" mit einem
Bereich von 0.0 bis 1.0 und einem Anfangswert von 0.3.

```

Diese Version des Codes verwendet `other turtles-here with [user?]`, um die Schildkröten in derselben Position zu überprüfen und zu zählen, ob sie Nutzer sind. So sollte das Programm jetzt korrekt sein.

```

turtles-own [user?] to setup clear-all ;; setup-globals setup-turtles reset-
ticks end ;to setup-globals ; set initial-users 10 ; Anzahl der
Anfangsnutzer ; set adoption-prob 0.3 ; Wahrscheinlichkeit, dass ein
Nicht-Nutzer ein Nutzer wird ;end to setup-turtles create-turtles 100 [ ;
Erstelle 100 Schildkröten setxy random-xcor random-ycor set user?
false ; Standardmäßig sind alle Schildkröten keine Nutzer set color blue
; Nicht-Nutzer sind blau ] ask n-of initial-users turtles [ ; Wählen Sie
zufällig eine Anzahl von Schildkröten als anfängliche Nutzer become-
user ] end to go ask turtles [ evaluate-adoption ] tick end to evaluate-
adoption if not user? [ ; Wenn die Schildkröte kein Nutzer ist let user-
neighbors count turtles in-radius 1 with [user?] if (user-neighbors > 0)
and (random-float 1 < adoption-prob) [ become-user ] ] end to become-
user set user? true set color green ; Nutzer sind grün end

```



Danke für das bereitgestellte Programm. Es scheint, dass die `initial-users` und `adoption-prob` Variablen auskommentiert wurden. Um das Programm auszuführen, müssen diese Variablen definiert sein. Hier ist eine korrigierte Version Ihres Codes:

```
netlogo
```

```
turtles-own [user?]
```

```
globals [initial-users adoption-prob]
```

```
to setup
  clear-all
  setup-globals
  setup-turtles
  reset-ticks
end
```

```
to setup-globals
  set initial-users 10          ; Anzahl der Anfangsnutzer
  set adoption-prob 0.3        ; Wahrscheinlichkeit, dass ein Nicht-
  Nutzer ein Nutzer wird
end
```

```
to setup-turtles
  create-turtles 100 [          ; Erstelle 100 Schildkröten
    setxy random-xcor random-ycor
    set user? false            ; Standardmäßig sind alle Schildkröten
    keine Nutzer
    set color blue              ; Nicht-Nutzer sind blau
  ]
  ask n-of initial-users turtles [ ; Wählen Sie zufällig eine Anzahl von
  Schildkröten als anfängliche Nutzer
    become-user
  ]
end
```

```
to go
  ask turtles [ evaluate-adoption ]
  tick
end
```

```
to evaluate-adoption
  if not user? [                ; Wenn die Schildkröte kein Nutzer ist
    let user-neighbors count turtles in-radius 1 with [user?]
    if (user-neighbors > 0) and (random-float 1 < adoption-prob) [
      become-user
    ]
  ]
end
```

```
to become-user
  set user? true
  set color green              ; Nutzer sind grün
end
```

In dieser Version des Codes sind `initial-users` und `adoption-prob` als globale Variablen definiert und mit ihren entsprechenden Werten initialisiert. Damit sollte das Programm nun wie erwartet funktionieren.

Ergänze das Programm, so dass die Nutzer umherlaufen



Um die Schildkröten im Programm umherlaufen zu lassen, können wir ihnen eine zufällige Bewegung pro Takt hinzufügen. Hier ist die aktualisierte Version Ihres Programms mit zufälliger Bewegung:

```
netlogo
```

```
turtles-own [user?]
```

```
globals [initial-users adoption-prob]
```

```
to setup
  clear-all
  setup-globals
  setup-turtles
  reset-ticks
end
```

```
to setup-globals
  set initial-users 10          ; Anzahl der Anfangsnutzer
  set adoption-prob 0.3        ; Wahrscheinlichkeit, dass ein Nicht-
  Nutzer ein Nutzer wird
end
```

```
to setup-turtles
  create-turtles 100 [          ; Erstelle 100 Schildkröten
    setxy random-xcor random-ycor
    set user? false            ; Standardmäßig sind alle Schildkröten
    keine Nutzer
    set color blue              ; Nicht-Nutzer sind blau
  ]
  ask n-of initial-users turtles [ ; Wählen Sie zufällig eine Anzahl von
  Schildkröten als anfängliche Nutzer
    become-user
  ]
end
```

```
to go
  ask turtles [
    evaluate-adoption
    move          ; Fügt die Bewegungsfunktion hinzu
  ]
  tick
end
```

```
to evaluate-adoption
  if not user? [                ; Wenn die Schildkröte kein Nutzer ist
    let user-neighbors count turtles in-radius 1 with [user?]
    if (user-neighbors > 0) and (random-float 1 < adoption-prob) [
      become-user
    ]
  ]
end
```

```
to move      ; Die Bewegungsfunktion
```

```

    right random 360    ; Zufällige Drehung
    forward 1           ; Bewegung um einen Schritt nach vorne
end

to become-user
  set user? true
  set color green      ; Nutzer sind grün
end

```

In diesem aktualisierten Programm wird die Funktion `move` definiert, die jedem Nutzer erlaubt, sich zufällig um eine zufällige Anzahl von Grad zu drehen und dann einen Schritt vorwärts zu gehen. Die `move`-Funktion wird in jedem Schritt für jede Schildkröte aufgerufen, um ihre Bewegung zu aktualisieren.

schreibe das Programm in python



Hier ist das Programm in Python, das die Verbreitung der Nutzung eines neuen Angebots durch eine agentenbasierte Simulation simuliert:

python

```
import random
```

```
class Turtle:
```

```
    def __init__(self, x, y):
```

```
        self.x = x
```

```
        self.y = y
```

```
        self.user = False
```

```
        self.color = "blue" if not self.user else "green"
```

```
    def evaluate_adoption(self, turtles, adoption_prob):
```

```
        if not self.user:
```

```
            user_neighbors = sum(1 for turtle in turtles if turtle.user
and abs(turtle.x - self.x) <= 1 and abs(turtle.y - self.y) <= 1)
```

```
            if user_neighbors > 0 and random.random() < adoption_prob:
```

```
                self.become_user()
```

```
    def become_user(self):
```

```
        self.user = True
```

```
        self.color = "green"
```

```
    def move(self, max_x, max_y):
```

```
        self.x += random.choice([-1, 0, 1])
```

```
        self.y += random.choice([-1, 0, 1])
```

```
        self.x = max(0, min(self.x, max_x))
```

```
        self.y = max(0, min(self.y, max_y))
```

```
class Simulation:
```

```
    def __init__(self, num_turtles, initial_users, adoption_prob,
world_size):
```

```
        self.turtles = [Turtle(random.randint(0, world_size[0]),
random.randint(0, world_size[1])) for _ in range(num_turtles)]
```

```
        for turtle in random.sample(self.turtles, initial_users):
```

```
            turtle.become_user()
```

```
        self.adoption_prob = adoption_prob
```

```
        self.world_size = world_size
```

```
    def step(self):
```

```
        for turtle in self.turtles:
```

```
            turtle.evaluate_adoption(self.turtles, self.adoption_prob)
```

```
            turtle.move(self.world_size[0], self.world_size[1])
```

```
    def run(self, steps):
```

```
        for _ in range(steps):
```

```
            self.step()
```

```
            self.display()
```

```
    def display(self):
```

```
        for y in range(self.world_size[1] + 1):
```

```

        for x in range(self.world_size[0] + 1):
            turtle_at_position = next((turtle for turtle in
self.turtles if turtle.x == x and turtle.y == y), None)
            if turtle_at_position:
                print("G" if turtle_at_position.user else "B", end="
")
            else:
                print(".", end=" ")
        print()

# Parameter festlegen
num_turtles = 100
initial_users = 10
adoption_prob = 0.3
world_size = (20, 20)
simulation_steps = 20

# Simulation ausführen
simulation = Simulation(num_turtles, initial_users, adoption_prob,
world_size)
simulation.run(simulation_steps)

```

Dieses Python-Programm simuliert eine agentenbasierte Modellierung ähnlich wie das NetLogo-Programm. Es gibt Schildkröten (Agenten) in einem 2D-Raster, die sich zufällig bewegen und die Verbreitung der Nutzung eines neuen Angebots simulieren.

Ergänze das Programm um ein Plot mit der Anzahl Nutzer



Um ein Plot mit der Anzahl der Nutzer über die Zeit zu erstellen, können wir die **matplotlib**-Bibliothek verwenden. Hier ist das aktualisierte Programm mit einem Plot:



python

```
import random
import matplotlib.pyplot as plt

class Turtle:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.user = False
        self.color = "blue" if not self.user else "green"

    def evaluate_adoption(self, turtles, adoption_prob):
        if not self.user:
            user_neighbors = sum(1 for turtle in turtles if turtle.user
and abs(turtle.x - self.x) <= 1 and abs(turtle.y - self.y) <= 1)
            if user_neighbors > 0 and random.random() < adoption_prob:
                self.become_user()

    def become_user(self):
        self.user = True
        self.color = "green"

    def move(self, max_x, max_y):
        self.x += random.choice([-1, 0, 1])
        self.y += random.choice([-1, 0, 1])
        self.x = max(0, min(self.x, max_x))
        self.y = max(0, min(self.y, max_y))

class Simulation:
    def __init__(self, num_turtles, initial_users, adoption_prob,
world_size):
        self.turtles = [Turtle(random.randint(0, world_size[0]),
random.randint(0, world_size[1])) for _ in range(num_turtles)]
        for turtle in random.sample(self.turtles, initial_users):
            turtle.become_user()
        self.adoption_prob = adoption_prob
        self.world_size = world_size
        self.num_users_over_time = [initial_users]

    def step(self):
        for turtle in self.turtles:
            turtle.evaluate_adoption(self.turtles, self.adoption_prob)
            turtle.move(self.world_size[0], self.world_size[1])
        num_users = sum(1 for turtle in self.turtles if turtle.user)
        self.num_users_over_time.append(num_users)

    def run(self, steps):
        for _ in range(steps):
            self.step()
```

```

        self.plot()

    def plot(self):
        plt.plot(range(len(self.num_users_over_time)),
self.num_users_over_time)
        plt.xlabel('Time Steps')
        plt.ylabel('Number of Users')
        plt.title('Number of Users Over Time')
        plt.show()

# Parameter festlegen
num_turtles = 100
initial_users = 10
adoption_prob = 0.3
world_size = (20, 20)
simulation_steps = 20

# Simulation ausführen und Plot erstellen
simulation = Simulation(num_turtles, initial_users, adoption_prob,
world_size)
simulation.run(simulation_steps)

```

Dieses aktualisierte Programm verwendet die **matplotlib**-Bibliothek, um einen Plot der Anzahl der Nutzer über die Zeit zu erstellen. Der Plot zeigt, wie sich die Anzahl der Nutzer im Laufe der Simulation ändert.

ChatGPT kann Fehler machen. Überprüfe wichtige Informationen.