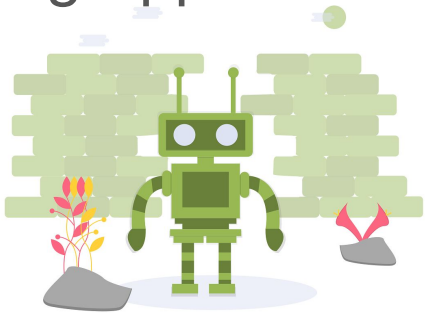
Entwicklung eines 3D RPG Videospiels mittels prozeduraler Inhaltsgenerieung und Deep Reinforcement Learning

Projektgruppe SoSe 2022

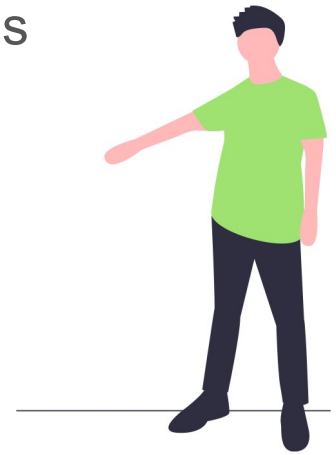


Marco Pleines & Nicolas Fischöder | Dortmund, 16.02.2022

Hi!

Wir sind Nicolas Fischöder & Marco Pleines

- Lehrstuhl 11 Algorithm Engineering
- Prof. Günter Rudolph
- Deep Learning
- Procedural Content Generation

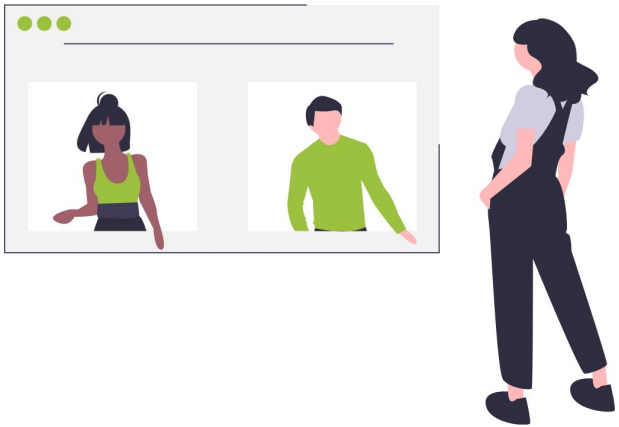


Wer seid Ihr?

- Bachelor wo gemacht?
- Thema Bachelorarbeit?

- Schon mal etwas mit …
  - … PCG …
  - … Unity ...
  - … Deep Learning …

  … gemacht?



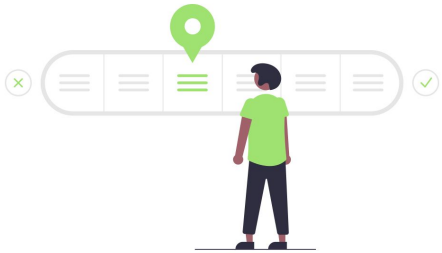
Agenda

Kurze Einführung

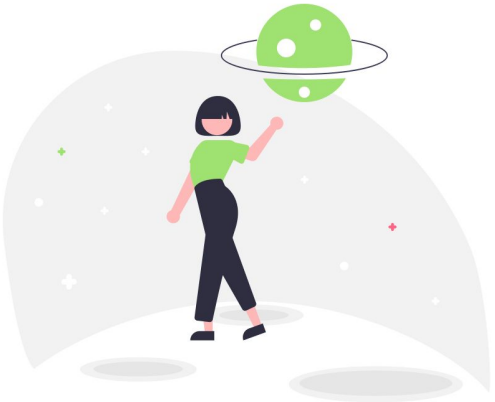
1. Grundlagen Deep Reinforcement Learning
2. Grundlagen Procedural Content Generation

Projektgruppe

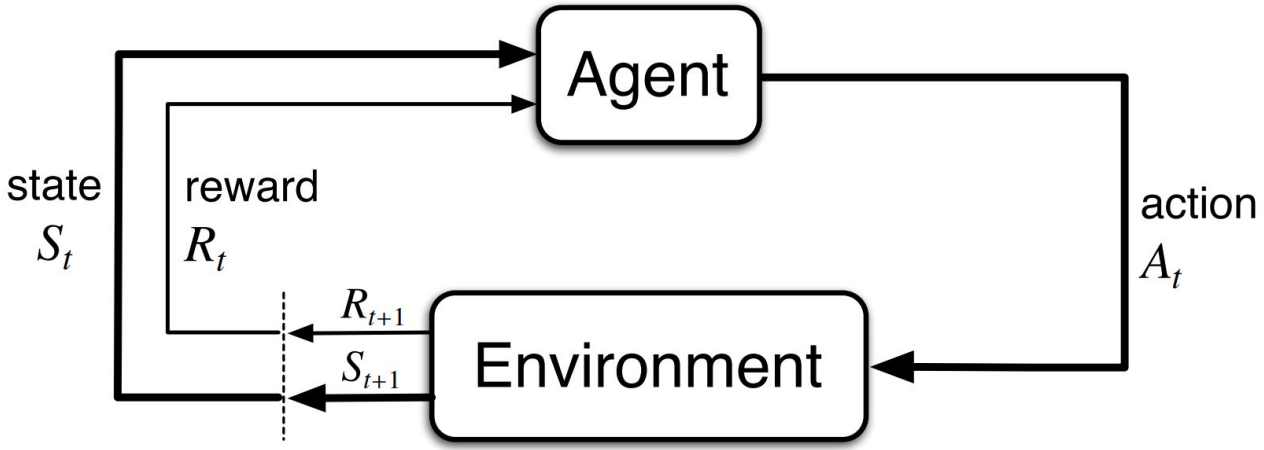
3. Vorhaben & Ziele
4. Seminar- und Praktikumsphase
5. Umfrage



1. Grundlagen DRL



Markov Decision Process



Transition / Experience tuple: S_t, A_t, R_t, S_{t+1}

Goal

- The agent shall autonomously learn a policy π
- Maximizing the cumulative reward

- Return:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- γ - discount factor

Value & Action-Value Function

Value Function

$$V_\pi(s) \doteq \mathbb{E}[G_t|S_t = s] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$

Q-Function or Action-Value Function

$$Q_\pi(s,a) \doteq \mathbb{E}[G_t|S_t = s, A_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right]$$

Advantage Function

$$A_\pi = Q_\pi(s, a) - V_\pi(s)$$

On-Policy vs. Off-Policy

- On-Policy (e.g. REINFORCE, PPO, SARSA)
  - Optimization can only utilize data that was produced by the current policy

- Off-Policy (e.g. Deep Q-Network, Q-Learning, SAC)
  - Optimization can be done with experiences that were produced by different policies (e.g. older ones)

Monte Carlo Sampling

- Entire trajectories are sampled before optimization
- Intention
  - The more we sample, the closer we approach the actual policy gradient
- REINFORCE utilizes Monte Carlo Sampling

Temporal-Difference Learning

- Optimization can be done right after taking one step
- Bootstrapping
  - Learned estimates are based on estimates
- Recursion
  - No need to sum rewards from R_0 until R_t (i.e. G_t)

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]$$

Q-Learning

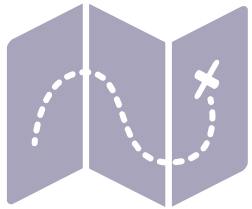
- Optimize Q-values
- α - learning rate
- Temporal-Difference Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

- Action selection ε-greedy:
  - Select (exploit) the most useful action or
  - sample (explore) an action given probability **ε**

Q-Learning

1. Initialize the Q-table
2. Set number of episodes ***E*** and max steps ***T***
3. **for** episode 0, …, ***E*** **do**
   a. Retrieve S_0
   b. **for** t = 0, …, ***T*** **do**
      i. Sample action A_t ε-greedy and step the environment
      ii. Optimize $Q(S_t, A_t)$
   c. **end for**
4. **end for**



Traditional Reinforcement Learning

- Tabular structure storing data for each state and each action (e.g. Q-Learning, SARSA)

- **Why does that not suffice?**

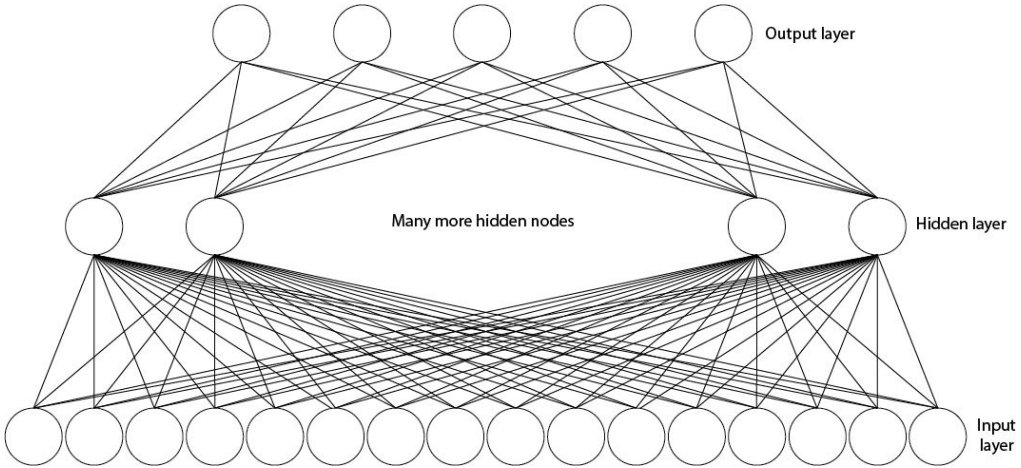
Traditional Reinforcement Learning

- Tabular structure storing data for each state and each action (e.g. Q-Learning, SARSA)

- **Why does that not suffice?**
  - Curse of Dimensionality
  - State space too big or continuous
  - Action space too big or continuous

Traditional Reinforcement Learning

- Artificial Neural networks ⇒ Deep Reinforcement Learning
- Potential to generalize



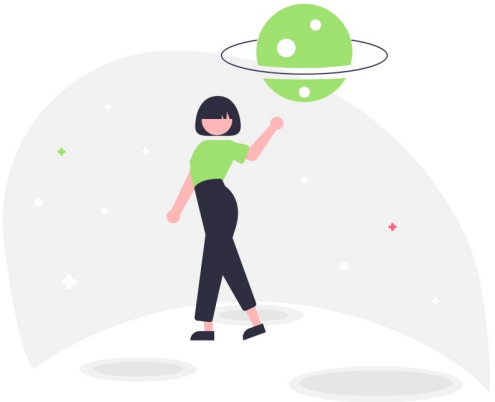
Deep Q Network (DQN)

- Action-Value Function is approximated by a neural net

- Episode Replay Buffer
  - Stores experience tuples
  - Sample mini batches

- Playing Atari with Deep Reinforcement Learning (2013)

2. Grundlagen PCG



Procedural Content Generation “Basics”

- Algorithmen-gestützte Generierung von Inhalten
- Problem: Kosten steigen proportional zu Anforderungen an Qualität
- Lösung: Prozeduren werden genutzt um Inhalte wie Modelle, Level, Sounds etc. (teil-)automatisiert zu generieren

Procedural Content Generation “Basics”

- Es gibt nicht den “*einen universellen Algorithmus*”
- Meist wird eine Kombination aus verschiedenen Algorithmen und Datenstrukturen benutzt
- zB.: Graphentheorie, Rauschfunktionen, Grammatiken, zelluläre-Automaten, neuronale Netze etc.

Procedural Content Generation “Basics”

- Prinzipiell kann jeder Algorithmus und jede Datenstruktur für PCG verwendet werden
- Einsatz von einzelnen Algorithmen ist vielfältig und unspezifisch
- Es existieren nur wenige öffentliche “Kochrezepte” für bestimmte Problemstellungen

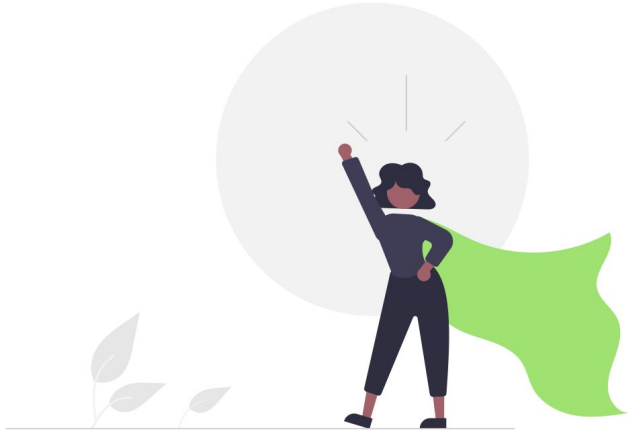
Procedural Content Generation “Basics”

- Graphen
- Rauschfunktionen (Perlin-Noise, Cellular-Noise etc.)
- KD-Bäume für rechteckige Unterteilungen von Arealen
- Grammatiken (L-Systeme, Shape-Grammars)

Procedural Content Generation “Basics”

- Algorithmen können in Code umgesetzt werden
- Es existiert aber auch Software, welche visuelles Scripting für PCG ermöglicht (SideFX Houdini)
- Runtime-generation vs. pre-authored

3. Vorhaben und Ziele



Prototyp 3D RPG Videospiel

(z.B. Hack & Slay)

Diablo

Elex

The Witcher

Aufgaben und Ziele der Projektgruppe

- Generierung von Spielleveln
- Evaluation der generierten level
- Generierung der Monster
- Fortbewegung der Monster
- Strategie bzw. Verhaltensweise der Monster

Generierung von Spielleveln

- Prozedurale Generierung von Spielleveln
  - z.B. KD-Trees, L-Systeme, Shape-Grammars
- Spiellevel enthält z.B. Boden, Wände, Spawn-Points für Props und NPCs
- Komplexität, Größe, Schwierigkeitsgrad der Level sollen parametrisierbar sein

Evaluation der generierten Level

Metriken

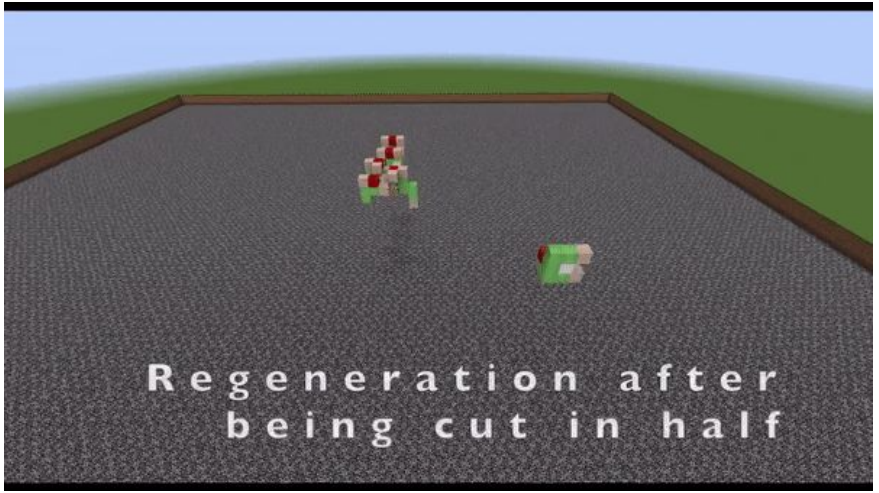
- Lösbarkeit der Level
- Schwierigkeitsgrad
- …

Ansätze

- Imitation Learning
- Deep Reinforcement Learning

Generierung der Monster

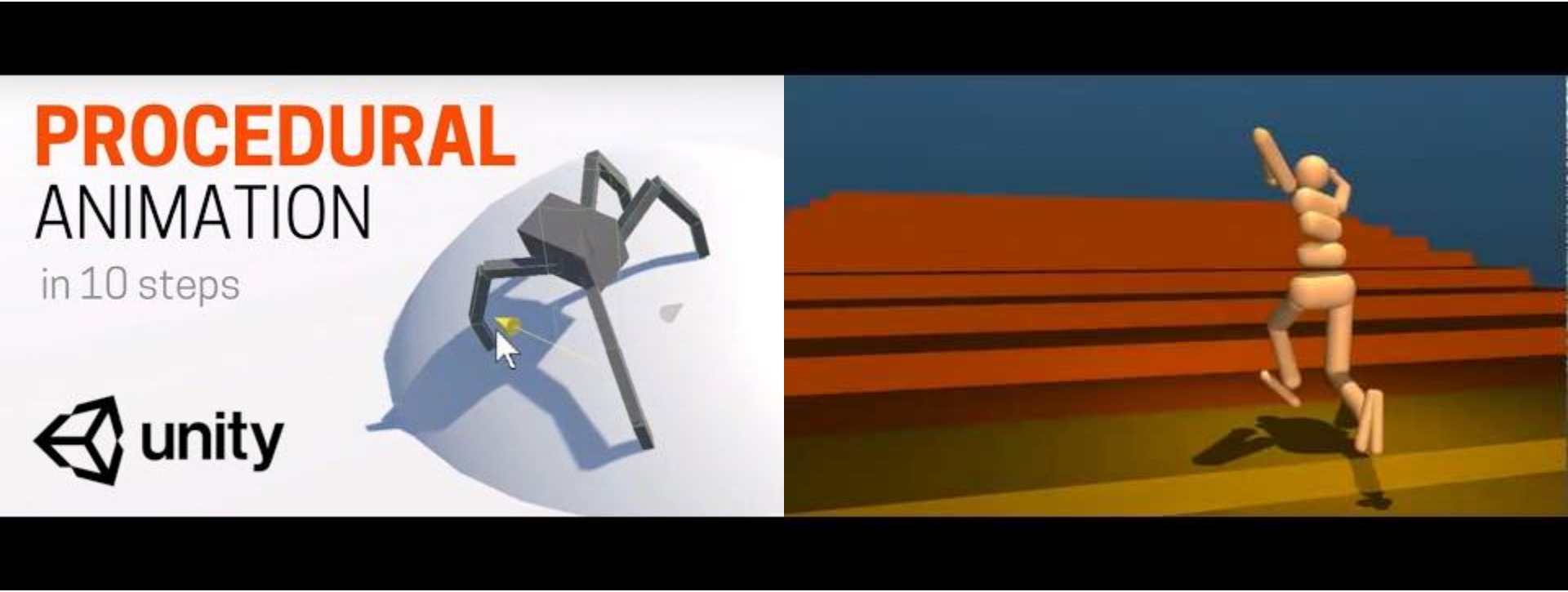
- Algorithmus-basierte Erstellung von NPCs
- Wichtig: Verwendung von Joints in Unity



Fortbewegung der Monster

- Animationen sollen nicht händisch erstellt werden
- Mittels DRL können die Monster lernen sich zu bewegen
- Der Agent wählt die Kräfte aus, welche auf seine Joints ausgeübt werden
- Je nach lösen einer spezifischen Aufgabe wird der Agent belohnt
- Ggf. nützlich: inverse Kinematik

Fortbewegung der Monster

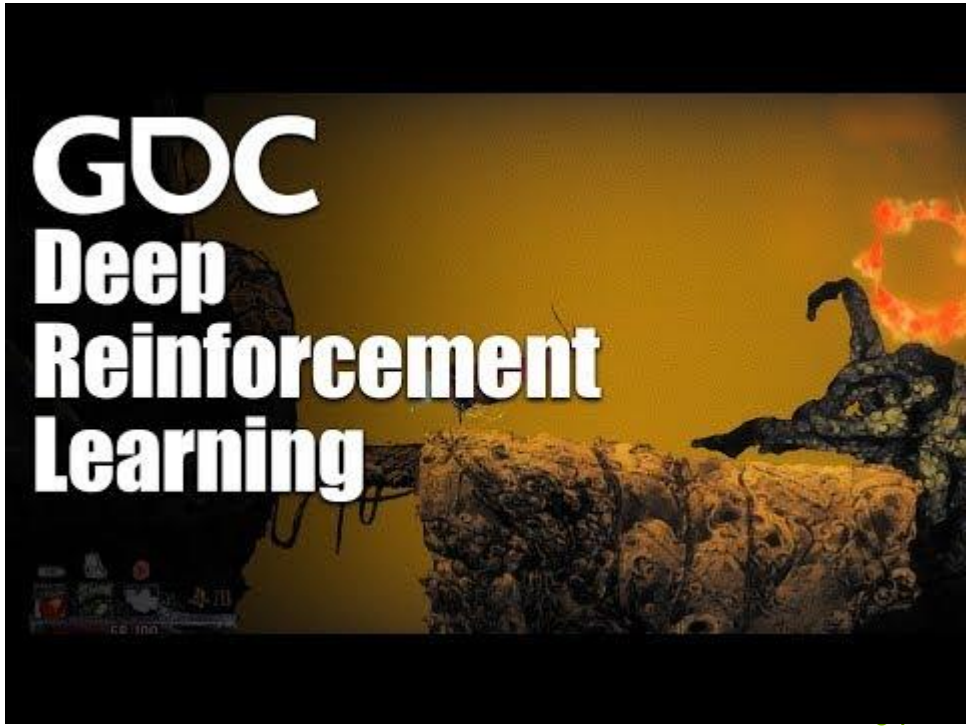


Fortbewegung der Monster

Fortbewegung der Monster

Beispiel in 2D

Source of Madness
(Carry Castle)



Strategie bzw. Verhaltensweise der Monster

Klassisch (u.a.):

- Behavior Trees
- State machines

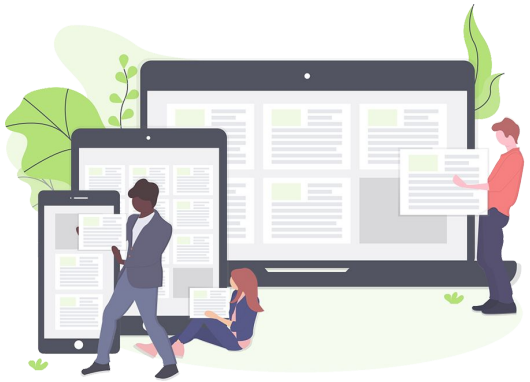
lernend:

- Imitation Learning
- Deep Reinforcement Learning

Verhaltensweise (high level)
- Patrol
- Attack
- Flee
- …

Fortbewegung (low level)
- Move forward
- Rotate left/right
- Attack
- …

4. Seminar- und Praktikumsphase



Marco Pleines & Nicolas Fischöder | Dortmund, 16.02.2022

Präsentationen im Seminar

- Dreierteams
- Wahl eines Präsentationsthemas
- Vorbesprechung mit den Betreuern
- Präsentationsdauer: 45 Minuten
- Diskussion: 10 Minuten
- Hands-On Session: 25 Minuten
- Diskussion: 10 Minuten
- Handout: 1 DIN A4 Seite

07. April 2022, 16 Uhr
14. April 2022, 16 Uhr
21. April 2022, 16 Uhr
28. April 2022, 16 Uhr

Praktikum

- Bearbeitung von Übungsaufgaben
- Betreuung während der Praktikumszeit (3h)

- Google Colab
- PC-Pool LS11 (remote)
- Live Share Visual Studio Code?

- Unity, ML-Agents
- PyTorch

08. April 2022, 9 Uhr
15. April 2022, 9 Uhr
22. April 2022, 9 Uhr
29. April 2022, 9 Uhr

Constructive Generation Methods

- Space Partitioning
- Zelluläre Automaten
- Evolutionäre zelluläre Automaten
- Generative Grammatiken

Creature Generation Methods

- Genetische Algorithmen
- Automatisches Rigging
- weitere ?

Proximal Policy Optimization (PPO)

- Policy Gradient
- Advantage Actor-Critic (A2C)
- Generalized Advantage Estimation
- Surrogate Objectives

Schulman et al. 2017. Proximal Policy Optimization Algorithms. Arxiv.

Soft Actor-Critic (SAC)

- Deep Deterministic Policy Gradients (DDPG)
- Twin-Delayed DDPG (TD3)
- Maximum Entropy
- Double Q-Learning
- Reparameterization Trick

Haarnoja et al. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. ICML: 1856-1865. Arxiv.

Weitere nützliche Quellen

https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

Prüfungsleistung

Richtlinien zur Durchführung von Projektgruppen **§7** & **§8**:

https://www.cs.tu-dortmund.de/nps/de/Studium/besondere_Lehrveranstaltungen/Projektgruppen/Rechtliches/index.html

⇒ Engagiert Euch!

5. Umfrage

Umfrage

- git?
- Google Colab, Jupyter Notebook?
- Unity?
- Deep Learning
  - PyTorch?
- Python IDE?

- Discord Server einrichten?

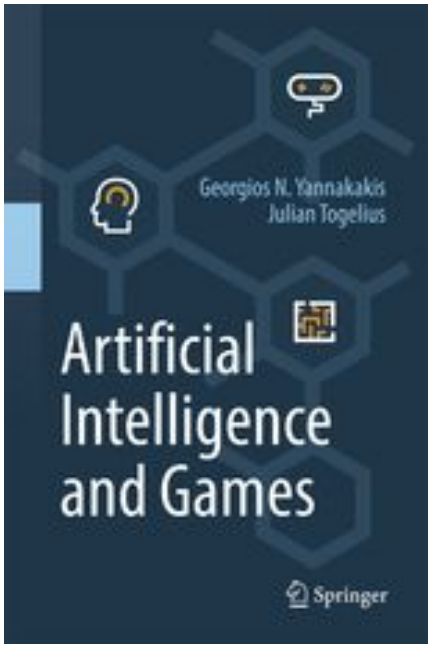
Literaturempfehlungen

Artificial Intelligence and Games

Yannakakis & Togelius

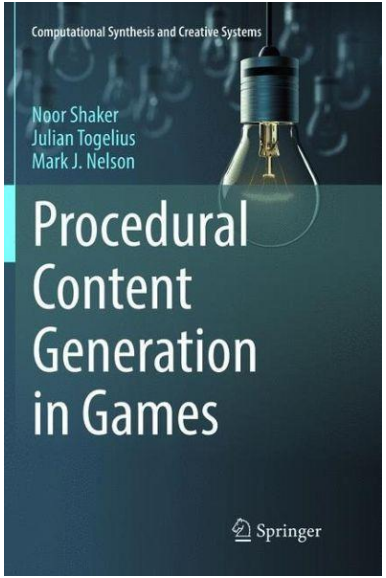
Springer

2018



Procedural Content Generation in Games

Shaker & Togelius
Springer
2018

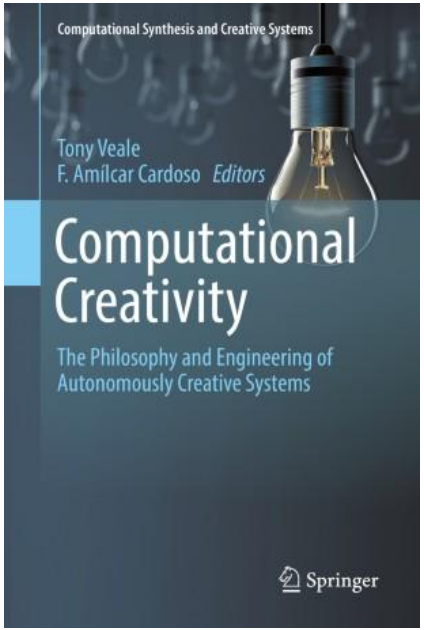


Computational Creativity

Veale & Cardoso

Springer

2019



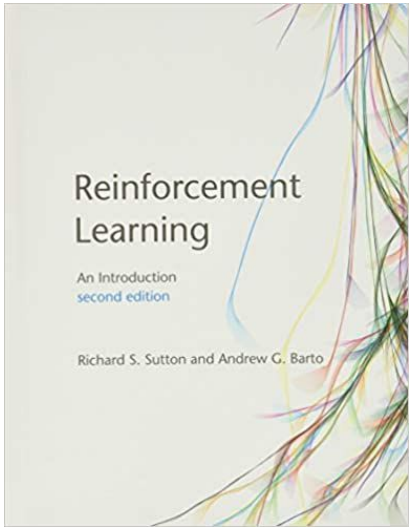
Reinforcement Learning

Sutton & Barto

2nd Edition

MIT Press

2018



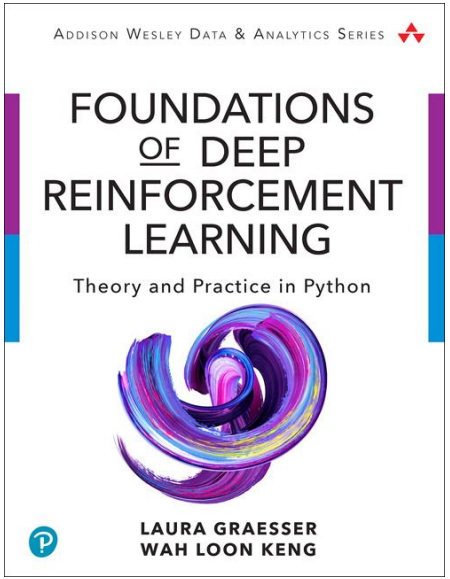
http://incompleteideas.net/book/the-book.html

Foundations of Deep Reinforcement Learning

Graesser & Keng

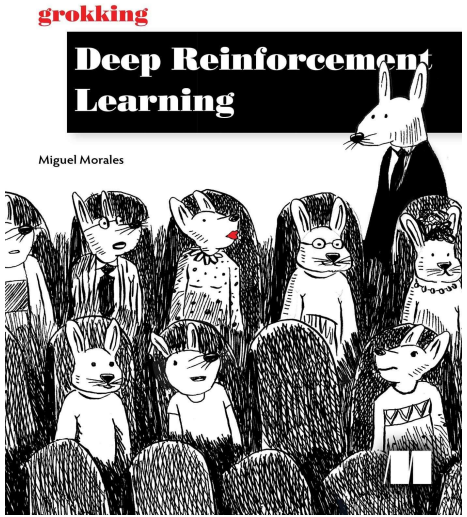
Pearson

2019



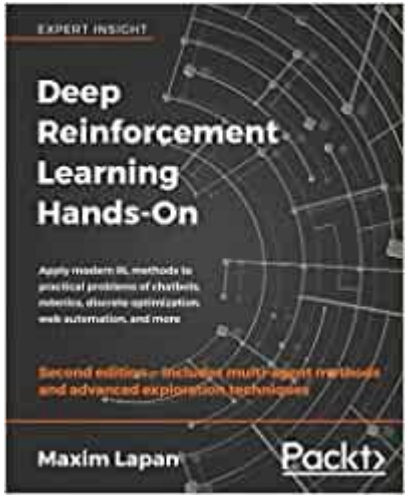
grokking Deep Reinforcement Learning

Miguel Morales
Manning
2020



Deep Reinforcement Learning Hands-On

Lapan
PacktPub
2018



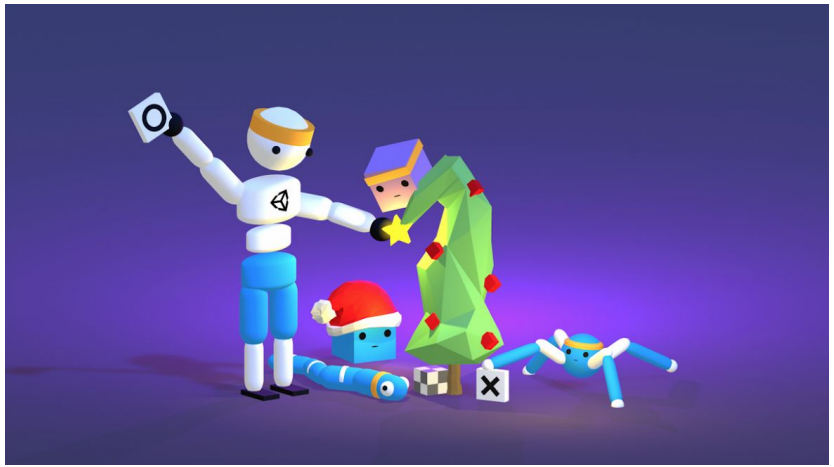
Machine Learning with Phil

https://www.youtube.com/channel/UC58v9cLitc8VaCjrcKyAbrw

Unity ML-Agents Toolkit

https://github.com/Unity-Technologies/ml-agents

https://blogs.unity3d.com/category/machine-learning/



Fragen?

Kontakt

Nicolas Fischöder
nicolas.fischoeder@tu-dortmund.de

Marco Pleines
marco.pleines@tu-dortmund.de

Mailverteiler
pg649@ls11.cs.uni-dortmund.de