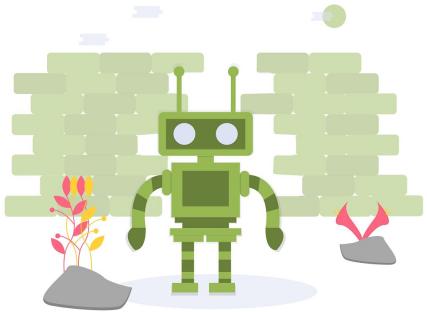
Deep Reinforcement Learning Environments
- Praktikum -

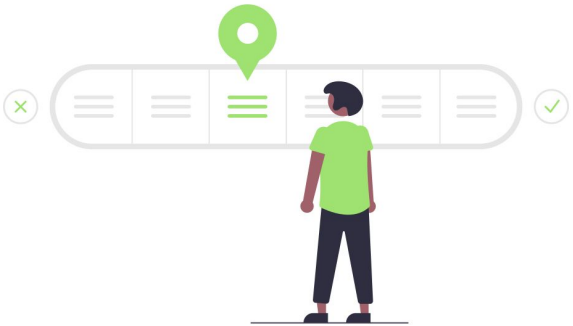
Projektgruppe 642 SoSe 2021



Marco Pleines | Dortmund, 16.04.2021

Agenda

1. Benötigte Software
2. Deep Reinforcement Learning Umwelten
   a. Dynamiken einer Umwelt
   b. Observationsraum
   c. Aktionsraum
   d. Belohnungsfunktion
3. Game Engine Unity
4. ML-Agents Toolkit



1. Benötigte Software

Benötigte Software

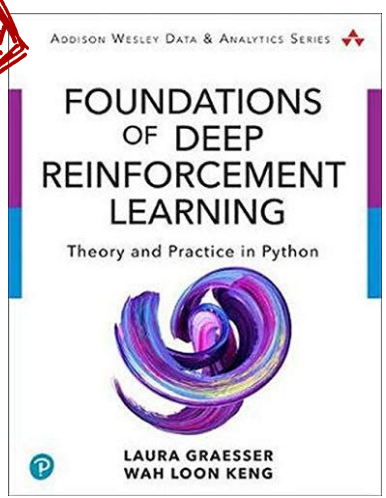
Installation von Unity, Python, Anaconda und IDE bitte vor dem Praktikum vornehmen!

- Unity 2019.4.* (Linux Installer)
- Python >= 3.6.1
- Anaconda (Alternative: virtualenv)
- Unity ML-Agents Toolkit Release 15

Das Praktikumsmaterial orientiert sich an dieser Software. Beachtet dies, falls ihr z.B. mit virtualenv abweicht.

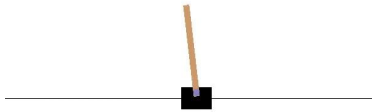
- Python IDE (z.B. Visual Studio Code, PyCharm, …)
- C# IDE (z.B. Visual Studio, MonoDevelop, ...)
- git Client (z.B. Sourcetree, GitHub Desktop, ...)

2. Deep Reinforcement Learning Umwelten



Seite 287
Kapitel IV
“Environment Design”

Deep Reinforcement Learning Umwelten



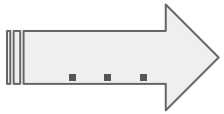
Dynamiken einer Umwelt

- Was ist die Aufgabe des Agenten in der Umwelt?
- Wie ist die Umwelt aufgebaut?
  - Statisch?
  - Dynamisch (z.B. prozedural)?
- Was passiert innerhalb einer Episode?
  - Gefahren?
  - Agierende Entitäten?
  - Veränderungen?

Dynamiken in Obstacle Tower

- Der Agent soll einen Turm von Ebene 0 bis 99 erklimmen
- Der Agent hat ein Zeitlimit
- Die Ebenen werden immer schl
- Prozedural generierte Level

Floor 15

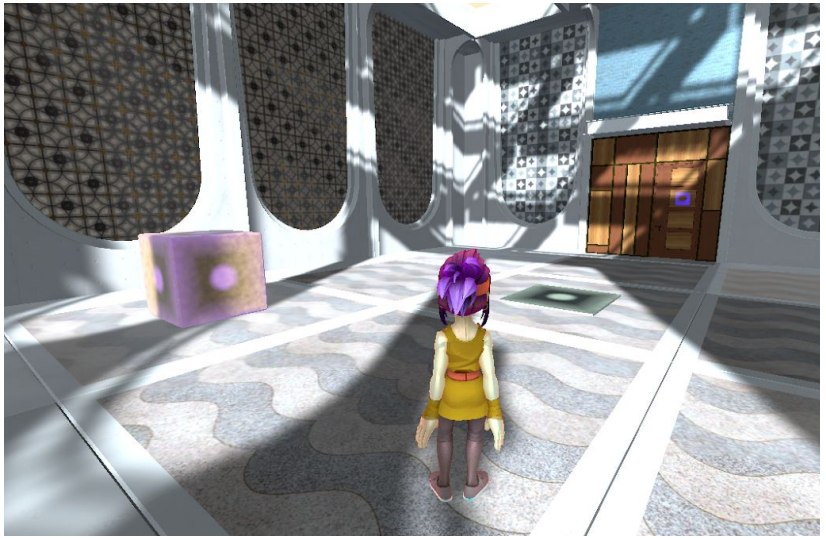


Floor 99

Dynamiken in Obstacle Tower

Schlüsselrätsel

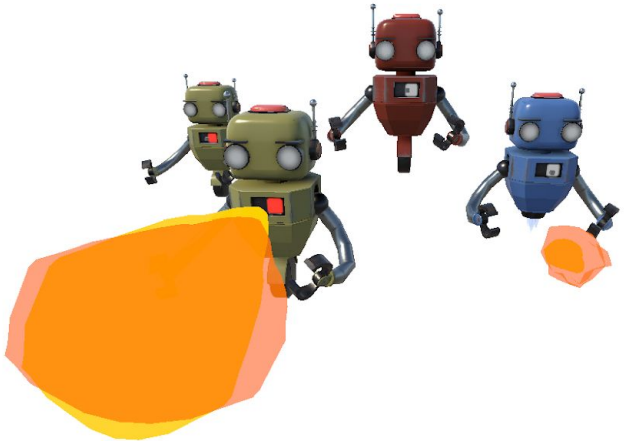
Sokoban-Rätsel



Dynamiken in Obstacle Tower

Gefahren

- Löcher im Boden
- Bewegende Arme und Plattformen
- Verschiedene Gegner

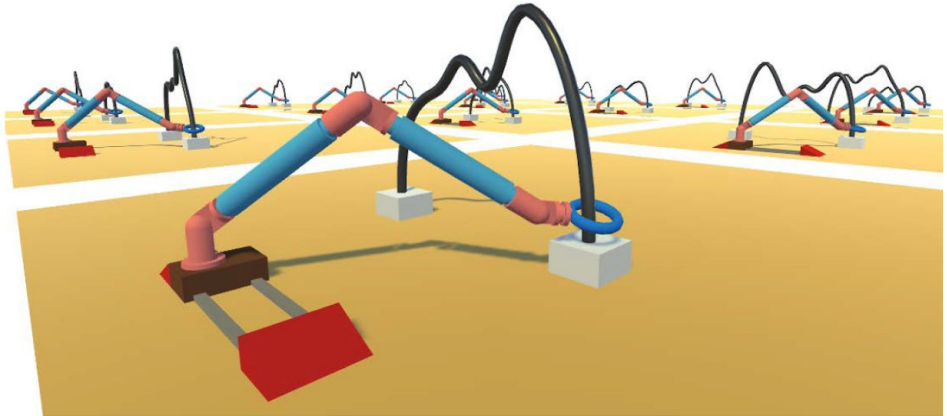


Dynamiken in Obstacle Tower

5 verschiedene Visual Themes

Dynamiken in Wireloop

- Der Agent (Roboterarm) muss in der Motorikschleife den Ring von A nach B manövrieren
- Der Draht darf nicht berührt werden
- Prozedural generierte Drähte



Observationsraum

- Welche Informationen stehen dem Agenten zur Verfügung um seine Umwelt wahrzunehmen?

Visuelle Observationen

(Textobservation)

Vektorobservationen

(Audioobservation)

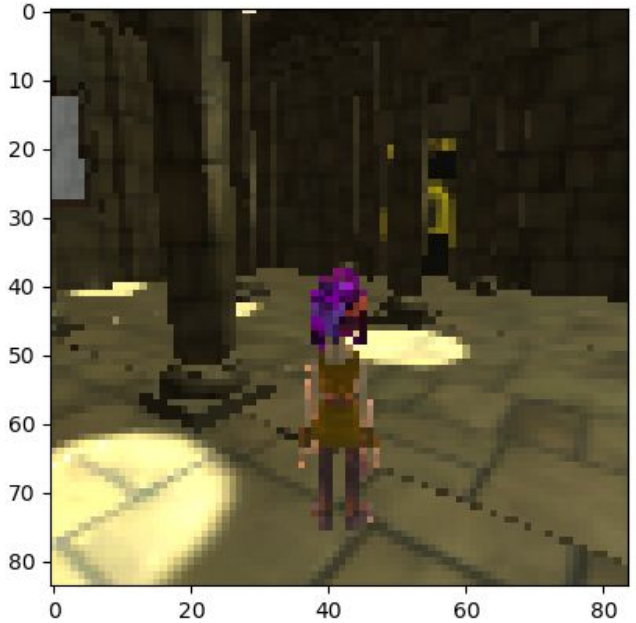
Observationsraum in Obstacle Tower

Visuell:

- z.B. RGB Bild (84x84x3)

Vektor:

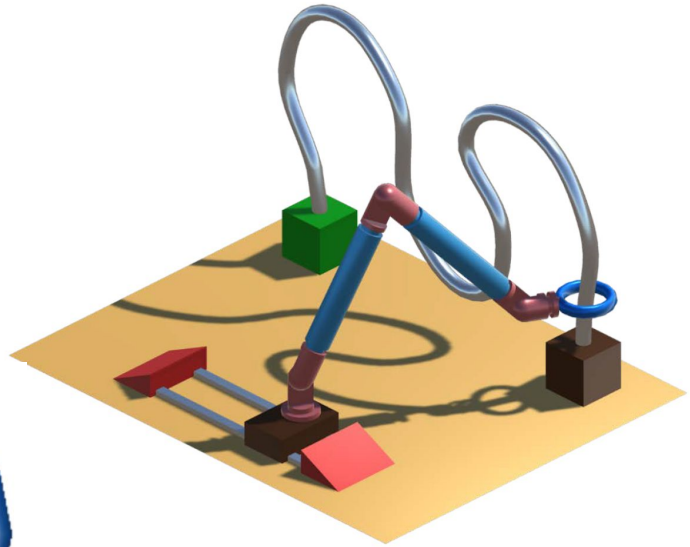
- besitzt Schlüssel?
- verbleibende Zeit

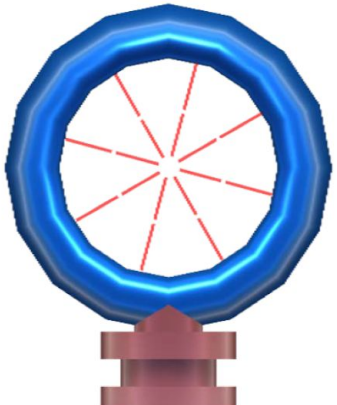


Observationsraum in Wireloop

Vektorobservation (78 Elementen):

- Drehmoment der Gelenke
- Position der Gelenke
- 24 Raycasts zum Messen der Abstände vom Inneren der Ringes





Aktionsraum

Arten von Aktionsräumen

- Kontinuierlich, Diskret, Multi-Diskret oder Hybrid

Kontinuierliche Aktionen

- Aktion wird durch einen reellen Wert dargestellt
  z.B. wie viel Gas gebe ich beim Autofahren?
  z.B. welcher Winkel soll das Lenkrad haben?

Aktionsraum

Diskrete Aktionen:

- Eine Aktion wird ausgeführt oder nicht
  z.B. betätigen einer Maustaste
  z.B. das Gaspedal um 5% weiter eindrücken

Multi-Diskret

- Beliebig viele diskrete Aktionsräume werden kombiniert
- Vorteil: Ausführung von mehreren Aktionen zeitgleich

Aktionsraum

Hybride Aktionsräume:

- Kombination aus diskrete und kontinuierliche Aktionen

- Eine diskrete Aktion wird gewählt und erhält zusätzliche Informationen durch die kontinuierliche Aktion

Hybride Aktion z.B. Skill Shot

- Diskrete Aktion:
  Löse Skill Shot A aus?

- Kontinuierliche Aktion:
  Winkel als Richtung

Aktionsraum in Obstacle Tower

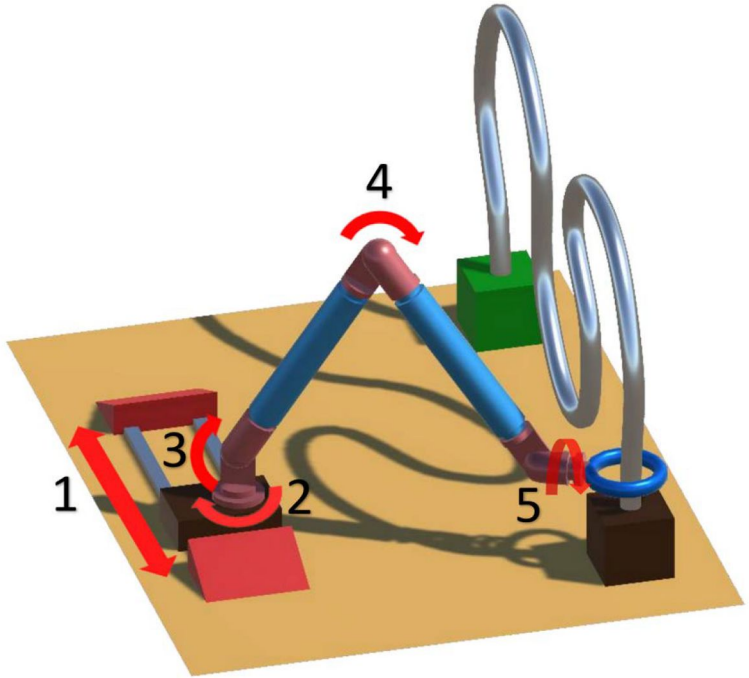
Multi-Diskret:

- Dimension A: No action, Move forward
- Dimension B: No action, Jump
- Dimension C: No action, Rotate left, Rotate right
- Dimension D: No action, Move left, Move right

Aktionsraum in Wireloop

Kontinuierlicher Aktionsraum:

1. Position auf der Schiene
2. Rotation an der Basis
3. Rotation des Unterarms
4. Rotation des Oberarms
5. Rotation des Rings



Belohnungsfunktion

- Feedback an den Agenten durch die Umwelt (extrinsische Belohnung)
- Agent belohnt sich selber (optionale intrinsische Belohnung)
- Positiver oder negativer Zahlenwert
- Beliebig oft ausschüttbar

Belohnungsfunktion in Obstacle Tower

- **+1.0** für das Erreichen der nächsten Ebene
- **+0.1** für das Öffnen einer Tür
- **+0.1** für das Einsammeln eines Schlüssels
  (bzw. für das Lösen des Sokoban-Rätsels)

Belohnungsfunktion in Wireloop

- **+1.0** für das Erreichen des Ziels
- **-1.0** für das Berühren des Drahts
- **-0.0001** für jeden benötigten Zeitschritt
- **Ringgeschwindigkeit * 0.01** (nur bei der richtigen Richtung), jeden Zeitschritt

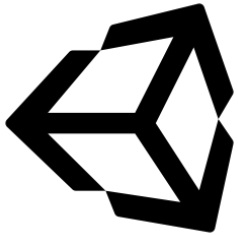
3. Game Engine Unity

Überblick

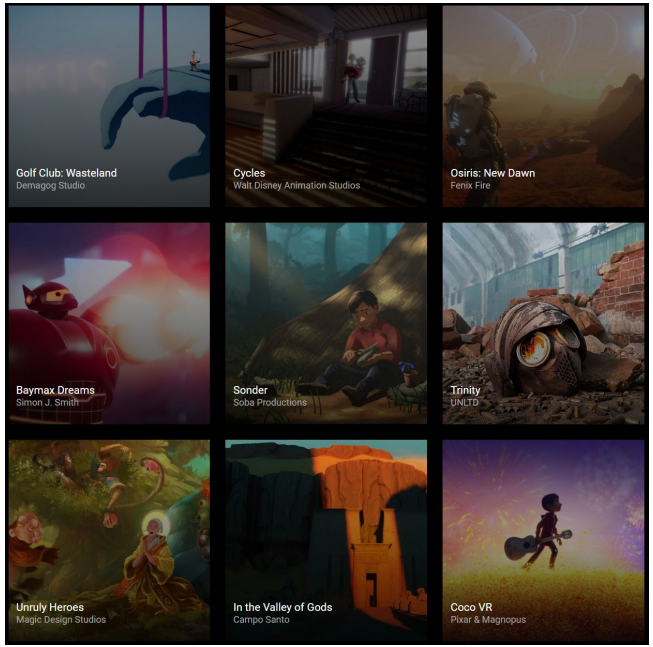
1. Was ist und was kann Unity?
2. Editor
3. Definitionen
4. Scripting
5. Interaktionen / Kommunikation
6. Ressourcen

Was ist Unity?

- Game Engine für 2D- und 3D-Anwendungen
- Programmiersprache C#
- Viele unterstützte Plattformen
- Großer Asset Store
- Kostenfrei bis zu einem bestimmten Umsatz



Was kann Unity?



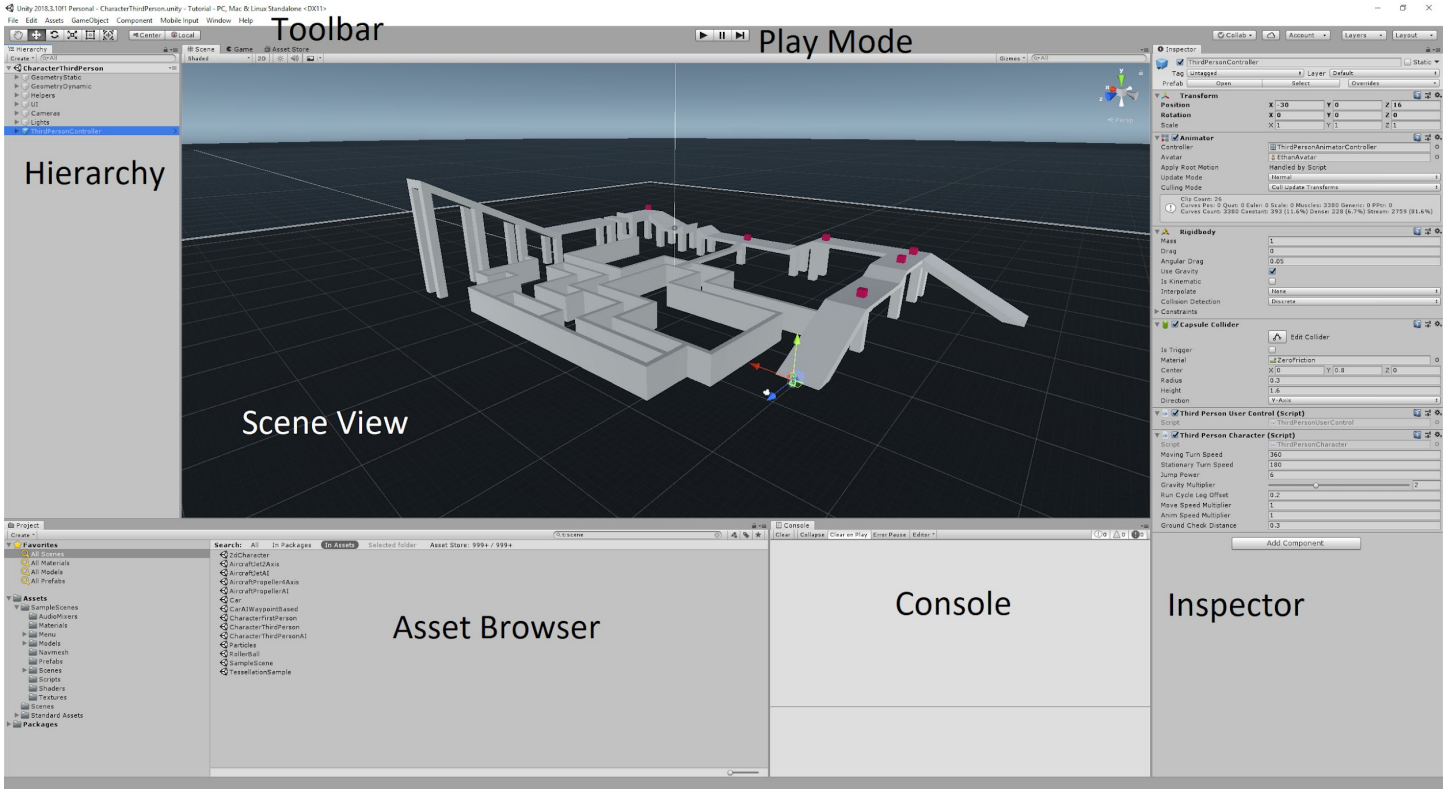
- Einige Erfolge
  z.B. ein aktuelles Wikinger-Survival-Spiel oder ein virtuelles Raumfahrtprojekt mit kleinen grünen Figuren …
- Teils auch schlechtes Image
  die Personal-Edition, die von Anfängern verwendet wird, blendet anfangs das Unity-Logo ein …
- 2D-, 3D-, VR-, AR- und weitere Anwendungen

Was kann Unity?

Rapid Prototyping

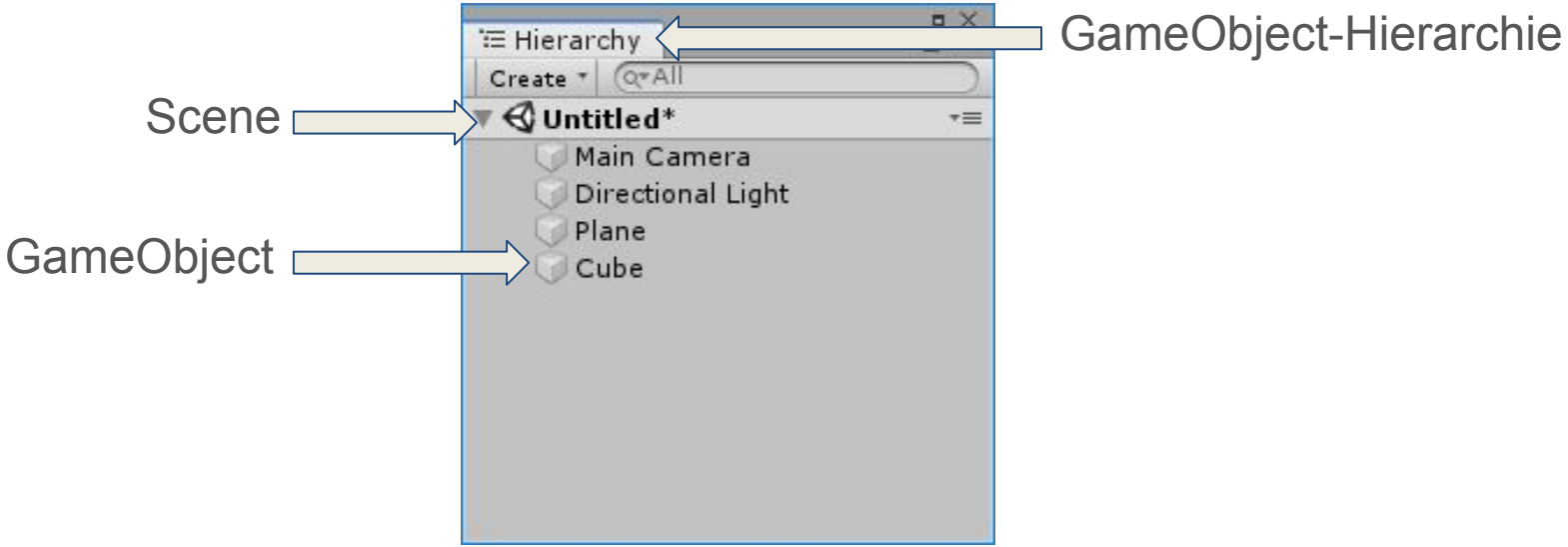
- https://itch.io/jam/brackeys-2
  - https://securas.itch.io/whitehearts
- https://globalgamejam.org/2019/games?title=&country=All&city=&tools=unity&diversifier=All&platforms=All

Editor


Toolbar
Play Mode
Hierarchy
Scene View
Asset Browser
Console
Inspector

Scene

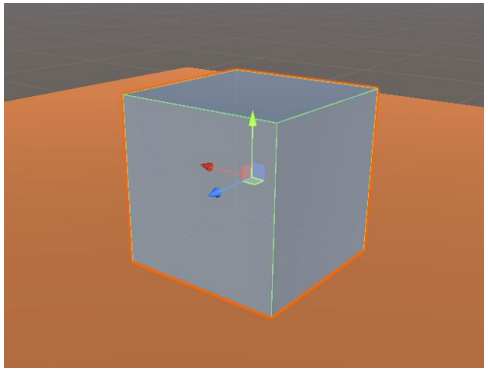
- Eine *Scene* setzt sich aus *GameObjects* zusammen



GameObject

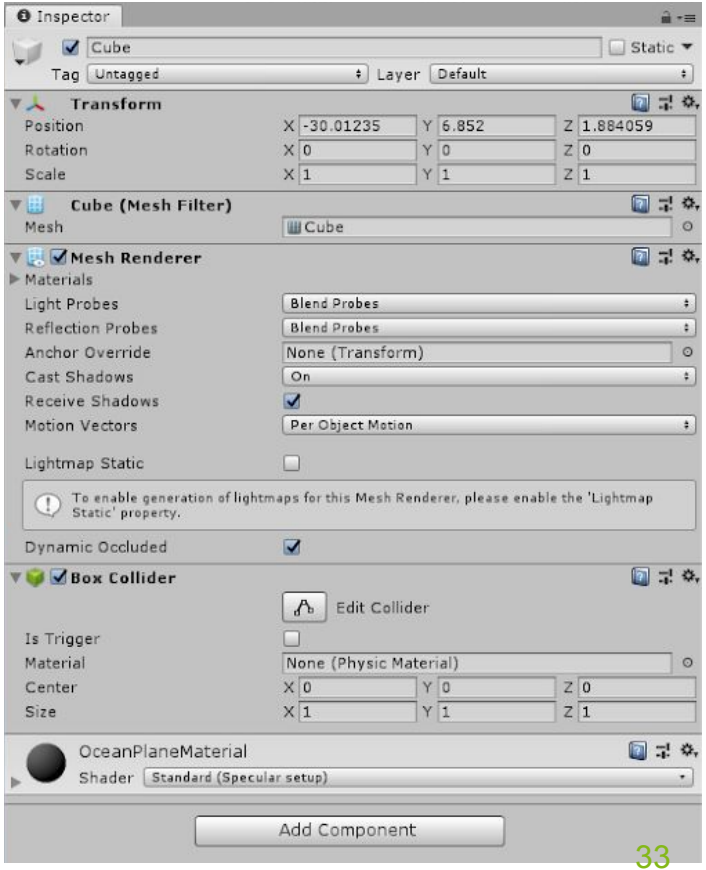
- Kernelement in Unity
- Ein GameObject hat einen Namen, *Layer* und *Tag*

- GameObjects bestehen aus *Komponenten* und erhalten dadurch ihre Eigenschaften und ihr Verhalten



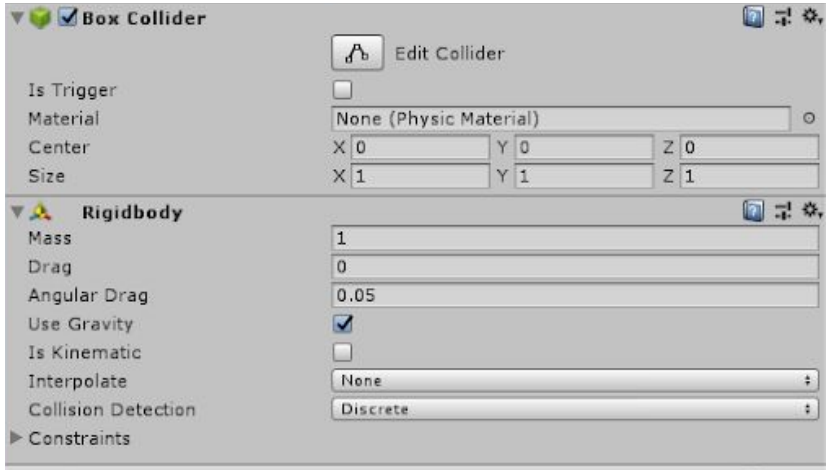
Components

- “Features” von GameObjects
- Jedes GameObject hat eine
- *Transform*-Komponente
- 3D-Objekte haben z.B. einen *Mesh Renderer* und ggf. einen *Collider*



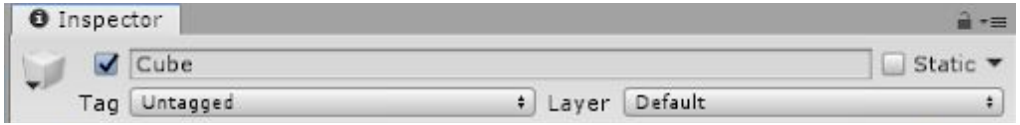
Physikkomponenten

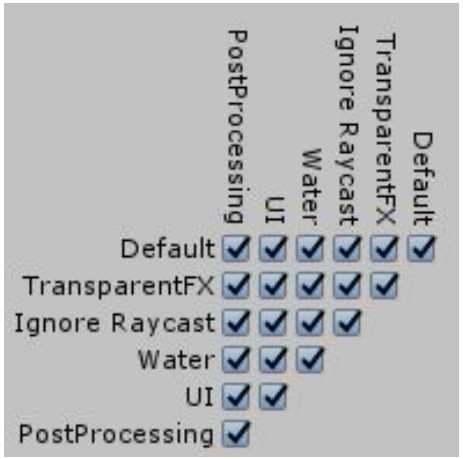
- *Rigidbody* und *Collider* gehören
- zur Physik-Engine
- Unity unterstützt sowohl 3D- als auch 2D-Physik



Layer

- Kollisionserkennung in Abhängigkeit von Layers
- Betreffen auch Rendering von Objekten





Weitere Komponenten

Komponenten für…

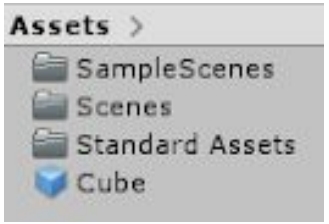
- Rendering
- 2D
- Audio
- AI
- UI
- ...

Weitere Beispiele:

- Kamera
- Audio Listener
- Animator
- Line Renderer
- Particle System
- Sprite Renderer
- …

Prefab

- Ein *Prefab* ist eine Blaupause für ein GameObject
- GameObjects werden als Prefab im Asset Browser abgespeichert
- Prefabs können zur Laufzeit (mehrfach) instanziiert werden



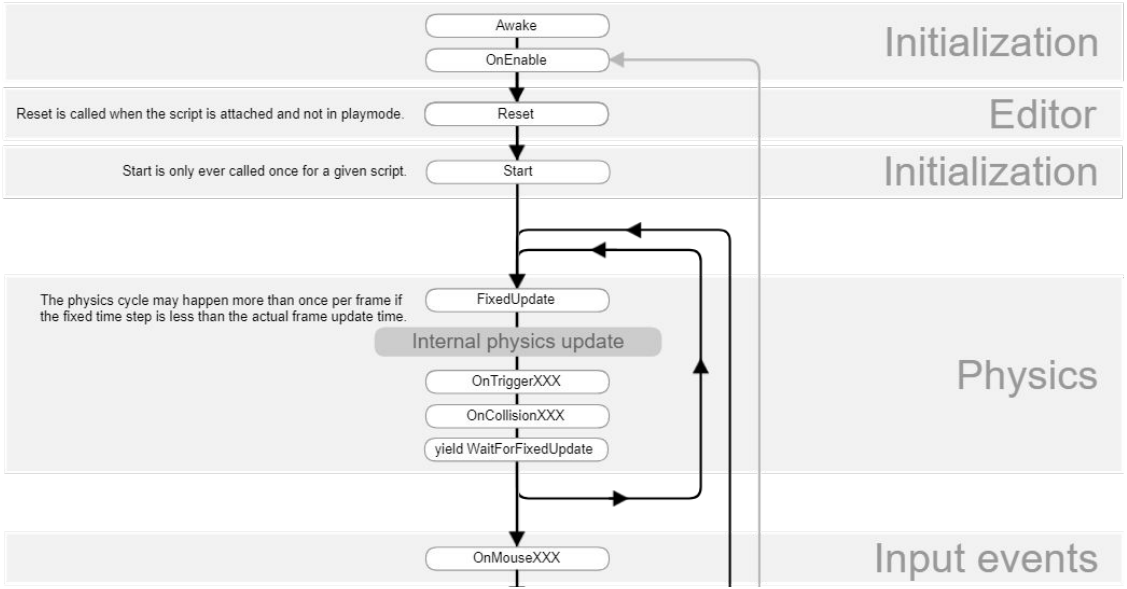
Scripting

- Programmiersprache: **C#**
  - Derzeit Version 8 für .NET Standard 2.0
- Im Kern ähnlich zu Java
  - Objektorientierung, Generics, Pakete / Assemblies, JVM / .NET Framework, …
- Bietet jedoch deutlich mehr Flexibilität und Kontrolle
  - *struct*, Erweiterungsmethoden, Operatorüberladung, *out*-Parameter, …

Scripting: MonoBehaviour

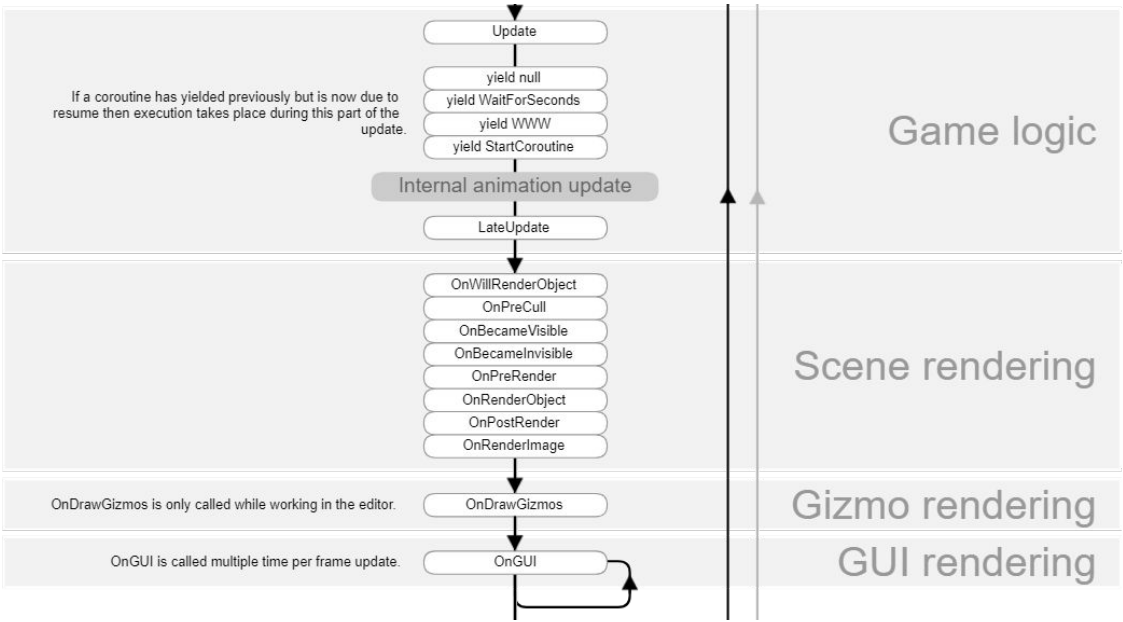
- Komponenten sind abgeleitete Klassen von *MonoBehaviour*
- Sämtliches Scripting geschieht initial über Komponenten

Scripting: Lifecycle



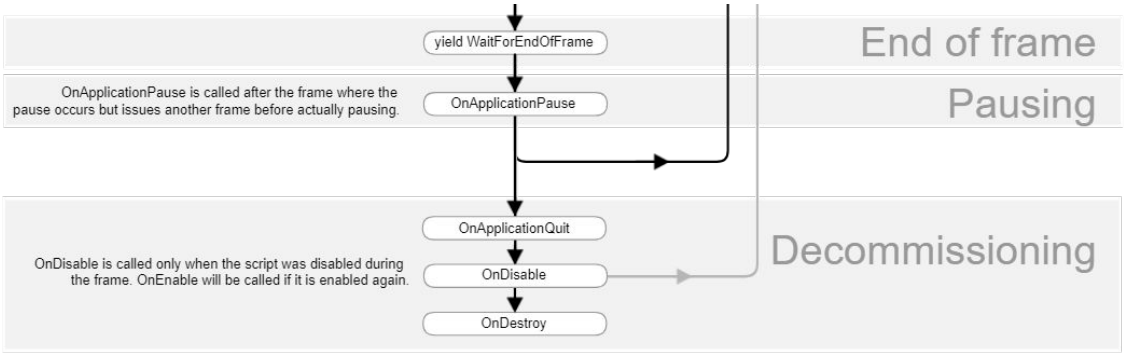
[Unity Manual]

Scripting: Lifecycle



[Unity Manual]

Scripting: Lifecycle



[Unity Manual]

Scripting: Events

- *Awake()*
  - Wird bei Konstruktion des Objekts ausgeführt
- *Start()*
  - Wird unmittelbar vor dem ersten Update ausgeführt
- *Update() und LateUpdate()*
  - Werden in jedem Frame aufgerufen
- *FixedUpdate()*
  - Wird in fixen Zeitintervallen aufgerufen (Physik)

Scripting Reference

Scripting: Events

- *OnEnable() / OnDisable()*
  - Wird bei der Aktivierung und Deaktivierung von einem Objekt aufgerufen
- *OnTrigger() / OnCollision()*
  - Wird bei einer Kollision aufgerufen
- *OnDestroy()*
  - Wird aufgerufen, wenn das Objekt zerstört wird

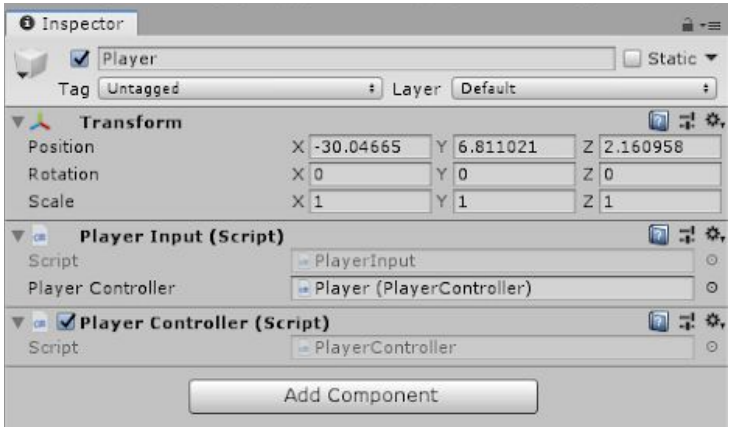
Scripting Reference

Interaktionen/Kommunikation

- Serialisierung von Variablen:
  - `public GameObject car;`
  - `[SerializedField] private float carSpeed = 1.0f;`

Interaktionen/Kommunikation

- Referenzierung von Komponenten per *Inspector*:



Interaktionen/Kommunikation

- Referenzierung von Komponenten per Script:

```
var player =
GameObject.FindGameObjectWithTag(“Player”);
var car = player.GetComponent<Car>();
```

Offizielle Ressourcen

- Unity User Manual
  - Relativ trocken, aber gute Textquelle
- Unity Scripting Reference
  - Wichtiges Nachschlagwerk
- Unity Learn
  - Schritt-für-Schritt-Tutorials mit persönlichem Fortschritts-Tracking (Account notwendig)
- Unity Standard Assets

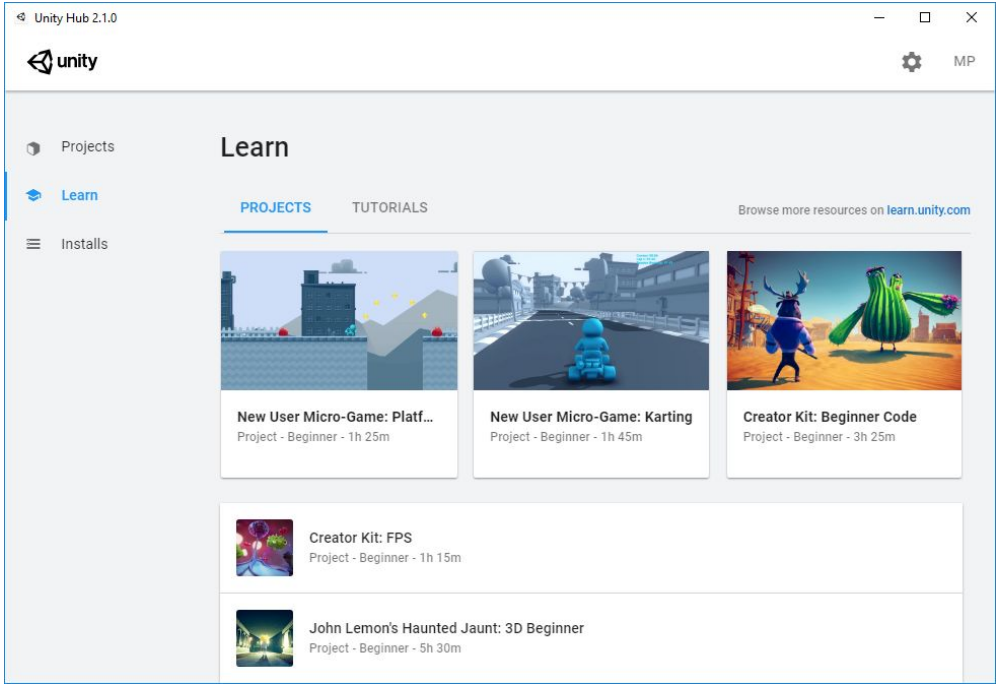
YouTube-Ressourcen

- Unity3D College
  - Detaillierte Unity-Einführung anhand der konkreten Entwicklung eines Spiels von Beginn an
- Hollistic3d
  - Einführung in einzelne Features (Videos zu Unity 5 ggf. veraltet)
- Brackeys
  - Programmierung, Game Design usw. (zeitlos, Kanal aber verwaist)
- quill18creates
  - Tutorials für vollständige Spieleprojekte

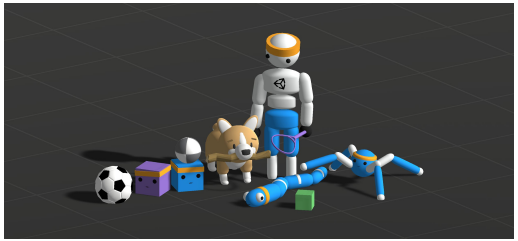
Weitere Ressourcen

- tutorialspoint
  - Tutorials zu den wichtigsten Themen zum Einstieg
- Catlike Unity C# and Shader Tutorials
  - Tiefgehende Scripting-Tutorials von einfachen Dingen bis zu fortgeschrittener Physik, Rendering, Noise und mehr
- JacksonDunstan.com
  - Fortgeschrittene C#-Programmierung auch in Unity, viele Benchmarks, leider kein Index: Suche nach “Unity”

Unity Learn


Unity Hub 2.1.0
unity
MP
Projects
Learn
Installs
Learn
PROJECTS
TUTORIALS
Browse more resources on learn.unity.com
New User Micro-Game: Platf...
Project - Beginner - 1h 25m
New User Micro-Game: Karting
Project - Beginner - 1h 45m
Creator Kit: Beginner Code
Project - Beginner - 3h 25m
Creator Kit: FPS
Project - Beginner - 1h 15m
John Lemon's Haunted Jaunt: 3D Beginner
Project - Beginner - 5h 30m

4. ML-Agents Toolkit



ML-Agents Toolkit (2017)

- Erstellen von Reinforcement Learning Umwelten
- Schnittstelle zwischen Unity und Python
- DRL Algorithmen (PPO, SAC, MA-POCA, self-play)
- Imitation Learning (Behavioral Cloning, GAIL)

Gym Interface

reset()

    return observation

step(action)

    return observation, reward, done, info

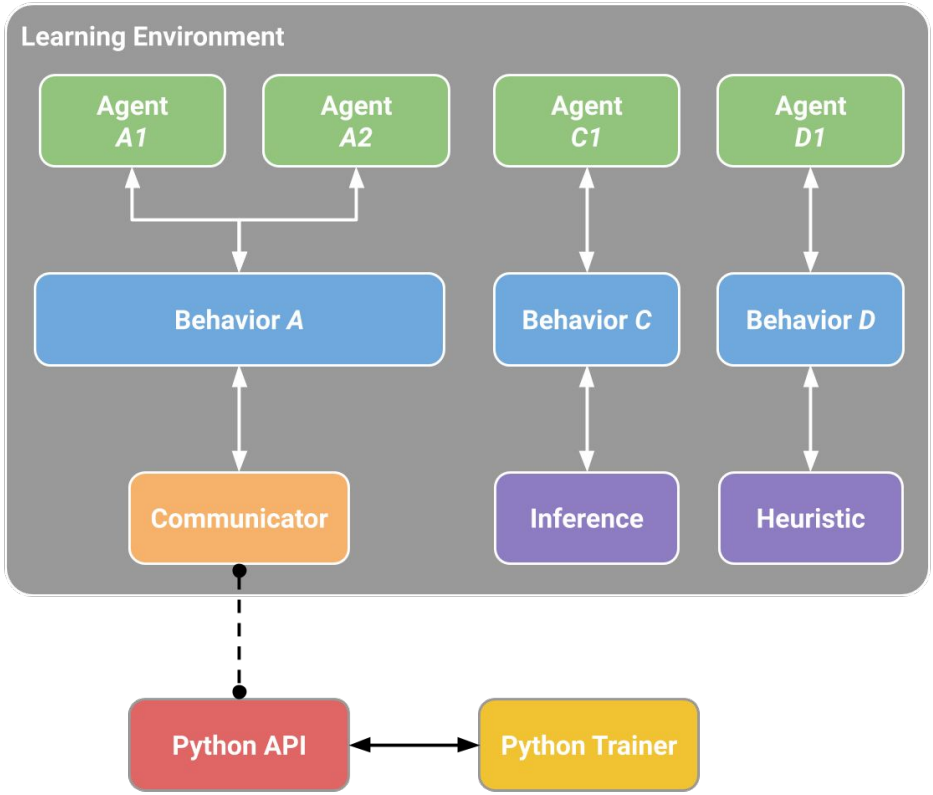
render(mode)

action_space

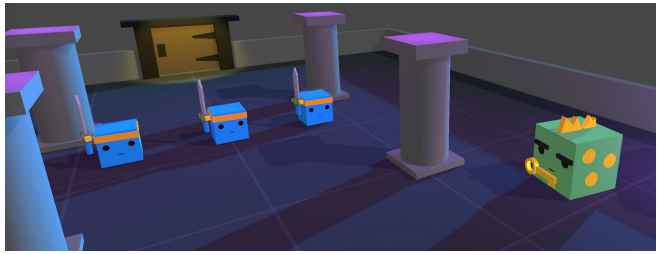
observation_space

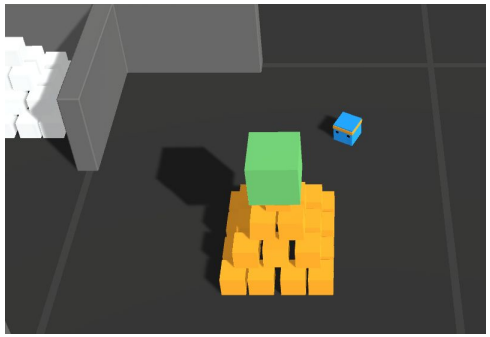
ML-Agents Toolkit

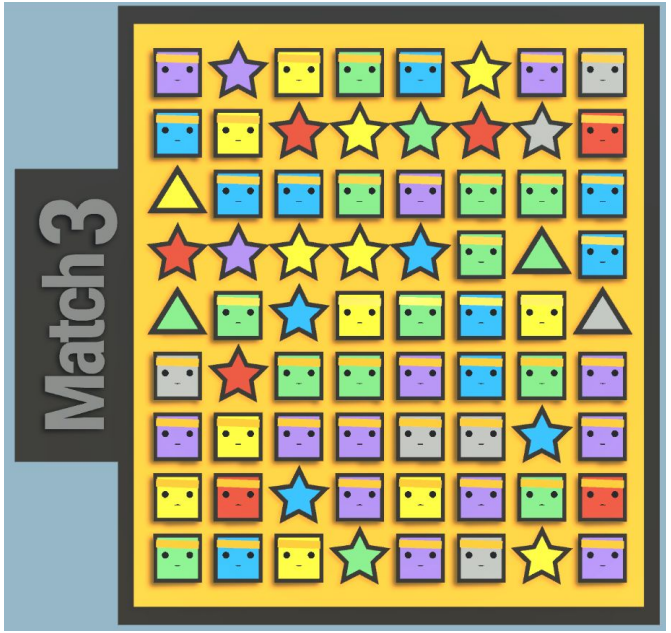
Overview



Beispielumwelten







Fragen?

Kontakt

Roman Kalkreuth
roman.kalkreuth@tu-dortmund.de

Marco Pleines
marco.pleines@tu-dortmund.de