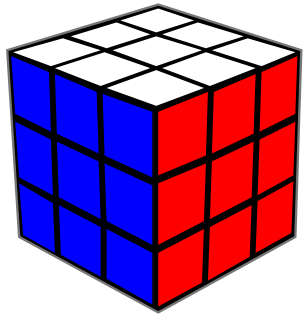
Unity

Game Development Unity
ITP

Rubiks Cube



Pacman

Dynablaster

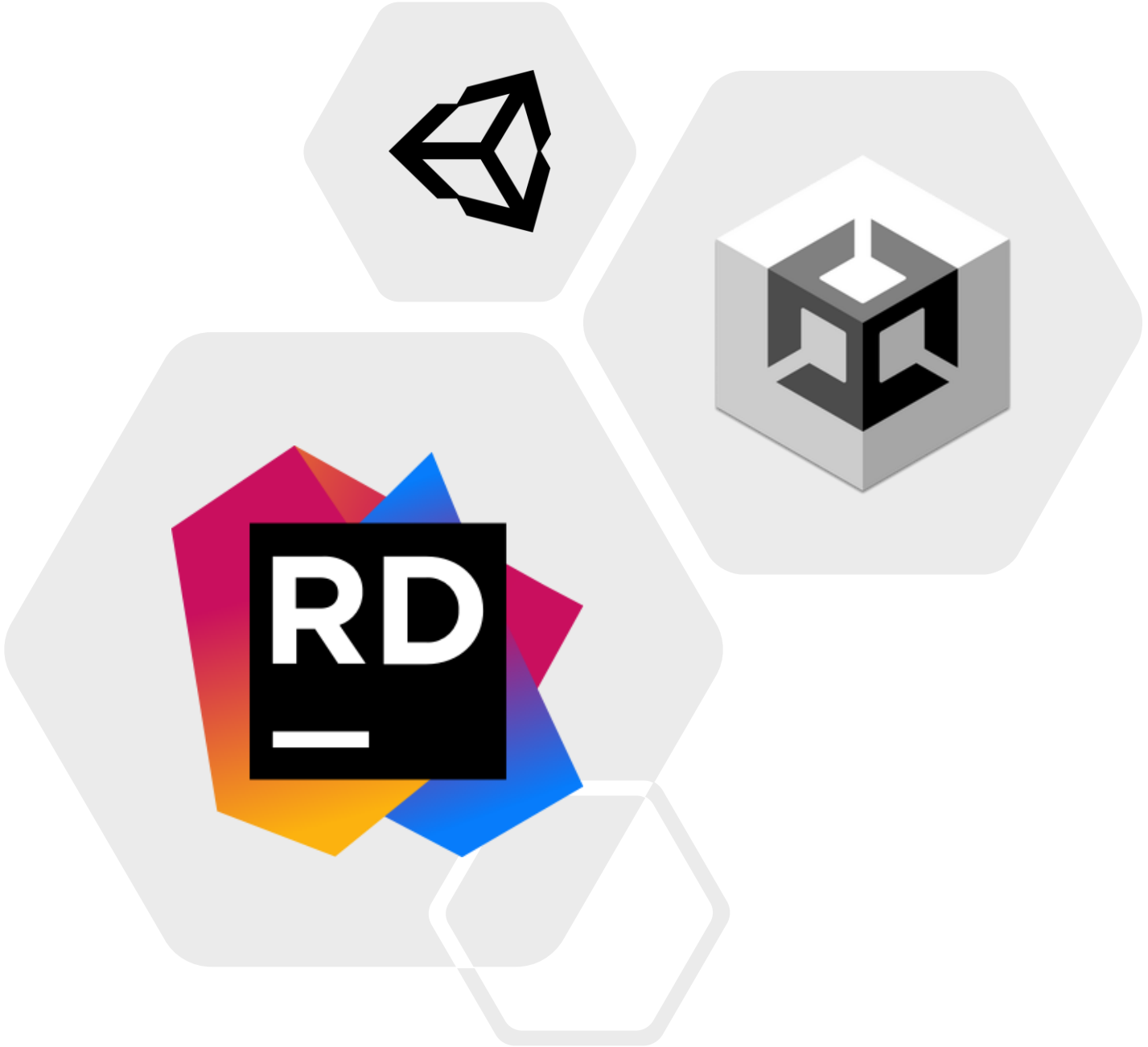
htl Hollabrunn

Übersicht

Technologien:	Unity / C# / NUnit
Projektthema:	Game Development Rubiks Cube, Pacman, Dynablast
Vorbesprechung:	Systemvorraussetzungen, erweiterter Inhalt, fachl. Vorraussetzungen

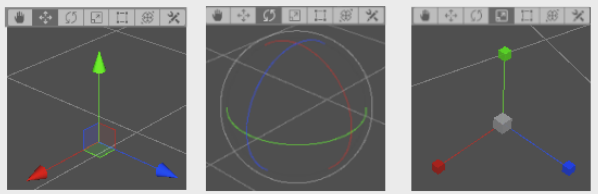
Übersicht

- Systemvorraussetzungen:
  - Funktionsfähiges Notebook!
  - Git-Zugang!
  - Installation folgender Software:
    - JetBrains Rider 2023.2.1 https://www.jetbrains.com/de-de/rider/
    - Unity Hub https://unity.com/download
    - Unity Editor 2022.3.9f1



Übersicht

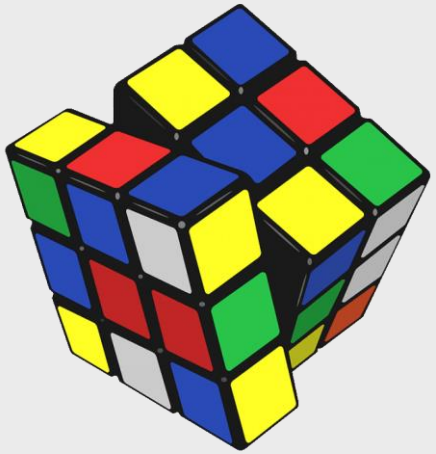
1. Woche:
   - Einführung in C#
   - Einführung Vektor-Rechnung
     - Translation
     - Rotation
     - Scaling
   - Coding Task mit Rider und Nunit

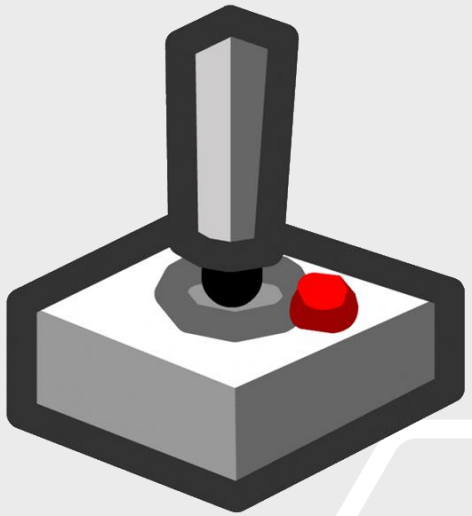


Übersicht

2. Woche:
   - Was ist … eine Game-Engine / eine Game-Loop?
   - Unity Editor
   - Spaces (Local vs Global)
   - GIT in Unity
   - Spritesheets

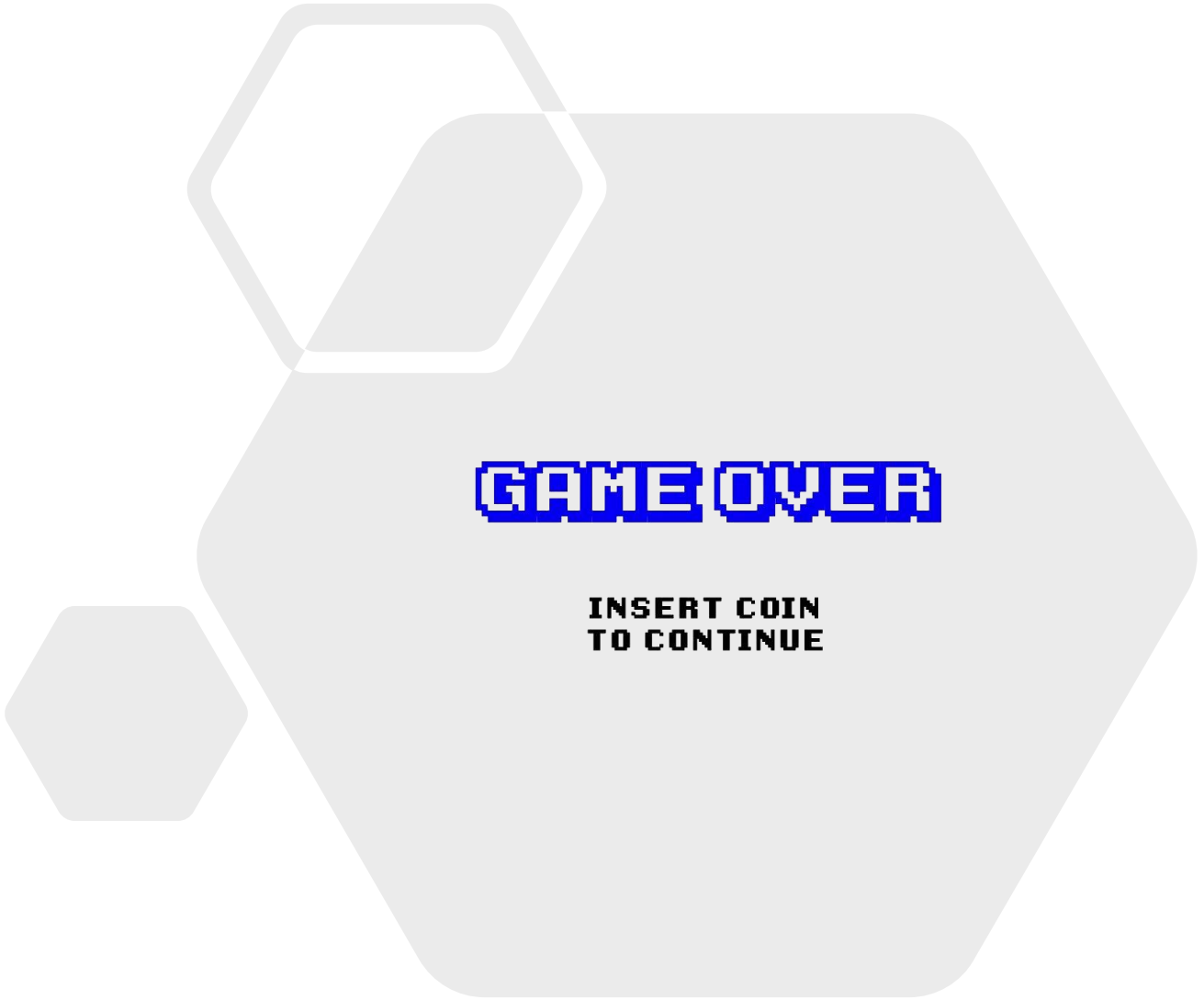
3. Woche:
   - Modellierung und Animation
   - Inputverarbeitung, Kamera





Übersicht

4. Woche:
   - Gamelogik
   - UI
   - Spielende erkennen
   - Neustart anbieten
   - Bonus: VR, Visual & Audio-Effects



BASED ON	.NET Core	.NET Framework
Open Source	.NET Core ist Open Source.	einige wenige Komponenten des sind Open Source
Cross-Platform	“build once, run anywhere” Windows, Linux, and Mac OS	Windows, Mono https://www.mono-project.com/
Application Models	Web, Windows Mobile, and Windows Store.	Desktop und Web-Applikationen
Installation	Installation unabhängig vom Betriebssystem	Paket für Windows OS
Support for Micro-Services and REST Services	Microservices und REST-API-Services	keine Microservices, REST API-Services.
Performance and Scalability	hohe Performance und Skalierbarkeit	weniger effizient im Vergleich zu .NET Core

htl Hollabrunn

BASED ON	.NET Core	.NET Framework
Compatibility	kompatibel mit Windows, Linux, and Mac OS.	Kompatibel mit Windows
Android Development	Xamarin für Mobile Development für iOS, Android, and Windows phones.	keine Unterstützung für Mobile Development
Packaging and Shipping	basierend auf Nugget Paketen	alle Bibliotheken werden als neue .NET Framework-Version ausgerollt
Deployment Model	Neue Versionen werden ausgerollt und können auf Systemen installiert werden. Kein Einfluss auf existierende Applikationen	Installation von ausgerollten .NET Frameworks wirken sich sofort auf laufende .NET Framework Applikationen aus

.NET Core

6.0.9

Latest runtime

6.0.401

Latest SDK

Supported channels

Channel	Support	Latest release	Latest release date	End of Life date
6.0	Current (LTS)	6.0.9	2022-09-13	2024-11-12
3.1	Maintenance	3.1.29	2022-09-13	2022-12-13

See all channels >

.NET Framework

4.8.1

Latest release

Recent releases

Version	Release date	CLR Version	Included in Windows	Included in Windows Server
4.8.1	2022-08-09	4	-	-
4.8	2019-04-18	4	10 May 2019 Update	2022
4.7.2	2018-04-30	4	10 April 2018 Update (Version 1803)	version 1803

Unity

htl Hollabrunn

C# Versionen

- aktuellste Version C# 10 (stable)
- neueste Version C# 11 (preview)

- Auswirkung auf sprachspezifische Features
- kommt C# 10 mit .NET 6.0

C# (c-sharp)

- https://dotnetbooks.blob.core.windows.net/ebooks/dotNET%20for%20Java%20Developers.pdf

- https://download.microsoft.com/download/D/E/E/DEE91FC0-7AA9-4F6E-9FFA-8658AA0FA080/CSharp%20for%20Java%20Developers%20-%20Cheat%20Sheet.pdf

UnitTests with NUnit

```
[TestFixture]
public class Vector3DTests
{
    private Vector3D _sut = new Vector3D();

    [SetUp]
    public void Init()
    {
        _sut = new Vector3D(0, 0, 0);
    }

    [Test]
    public void Vector3D_Add_ReturnsOkResult()
    {
        Assert.IsTrue(true);
    }
}
```

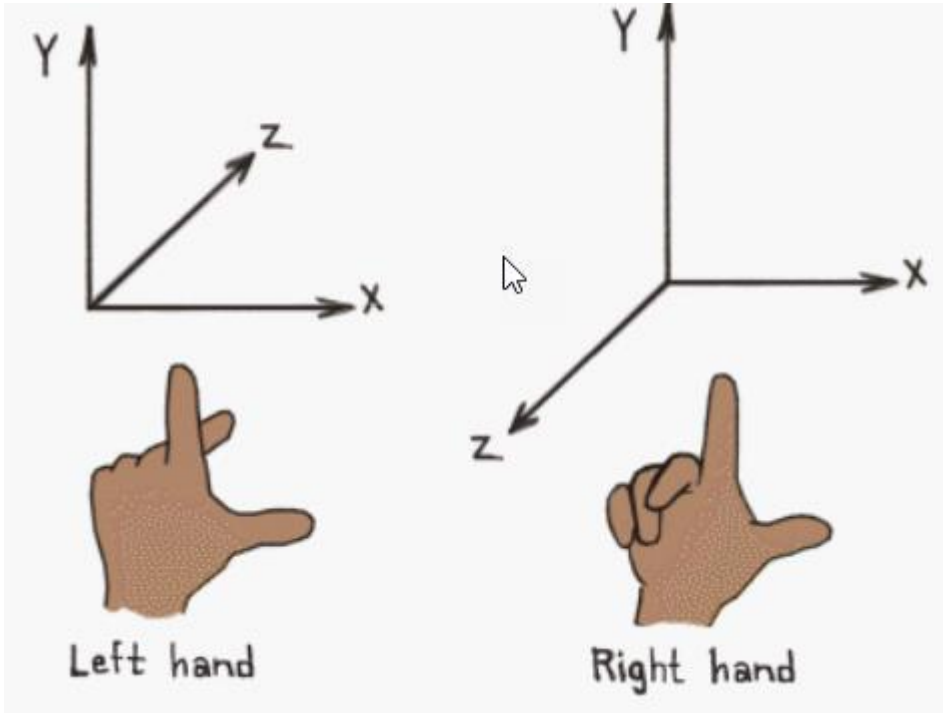
Documentation

- C# - https://docs.microsoft.com/en-us/dotnet/csharp/
- Unity - https://docs.unity3d.com/Manual/index.html
- Jetbrains Rider - https://www.jetbrains.com/help/rider/Introduction.html
- GIT LFS - https://docs.github.com/en/repositories/working-with-files/managing-large-files/installing-git-large-file-storage

Vektoren

Vektor im

- 2-dimensionalen Raum: $\begin{pmatrix} x \\ y \end{pmatrix}$
- 3-dimensionalen Raum: $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$



- x = Koordinate auf der x-Achse
- y = Koordinate auf der y-Achse
- z = Koordinate auf der z-Achse

Vektoren

Skalar

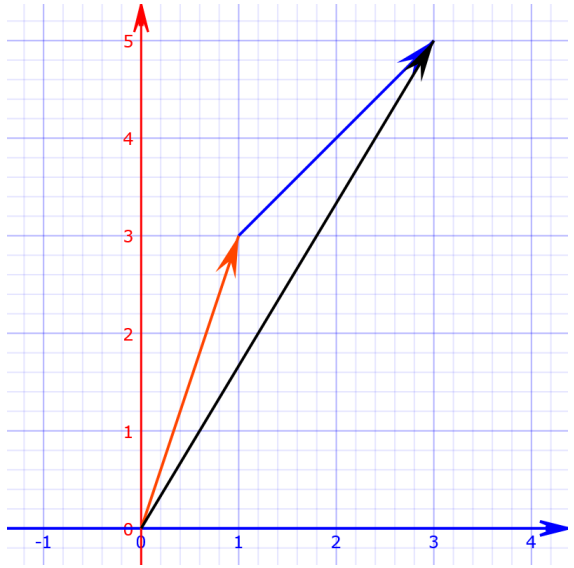
- Mathematische Größe, die allein durch Angabe eines Zahlenwertes charakterisiert ist
- Die Multiplikation eines Vektors mit einem Skalar heißt Skalarmultiplikation oder auch Skalierung.
- Der resultierende Vektor heißt “skalares Vielfaches des Ausgangsvektors”

Vektoren - Translation

$$x_0 = 1, \quad y_0 = 3$$
$$x_1 = 2, \quad y_1 = 2$$

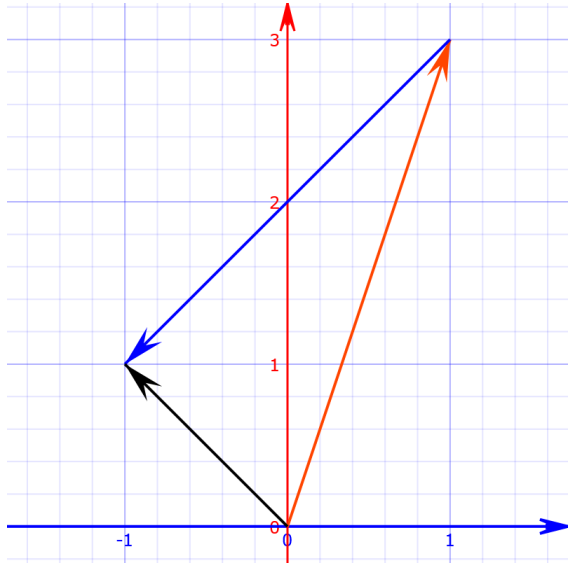
- Addition

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 + x_1 \\ y_0 + y_1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 3 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 + 2 \\ 3 + 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$



- Subtraktion

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 - x_1 \\ y_0 - y_1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 3 \end{pmatrix} - \begin{pmatrix} 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 - 2 \\ 3 - 2 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

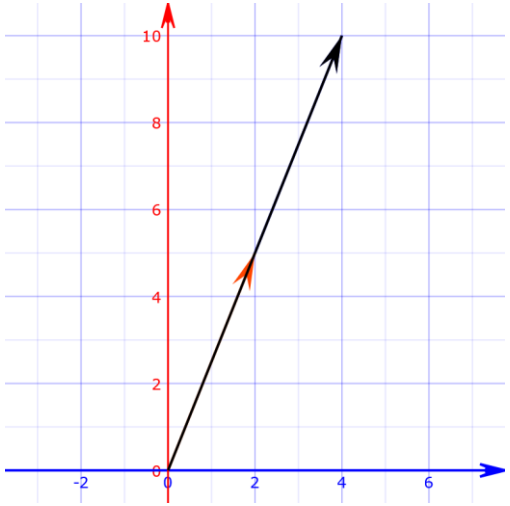


Vektoren - Skalierung

$x = 2, \qquad y = 5$

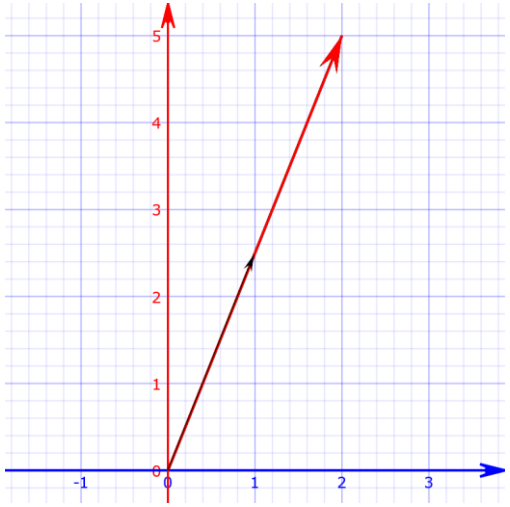
- Verdoppelung eines Vektors

$$2 \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \end{pmatrix} \qquad 2 \cdot \begin{pmatrix} 2 \\ 5 \end{pmatrix} = \begin{pmatrix} 4 \\ 10 \end{pmatrix}$$



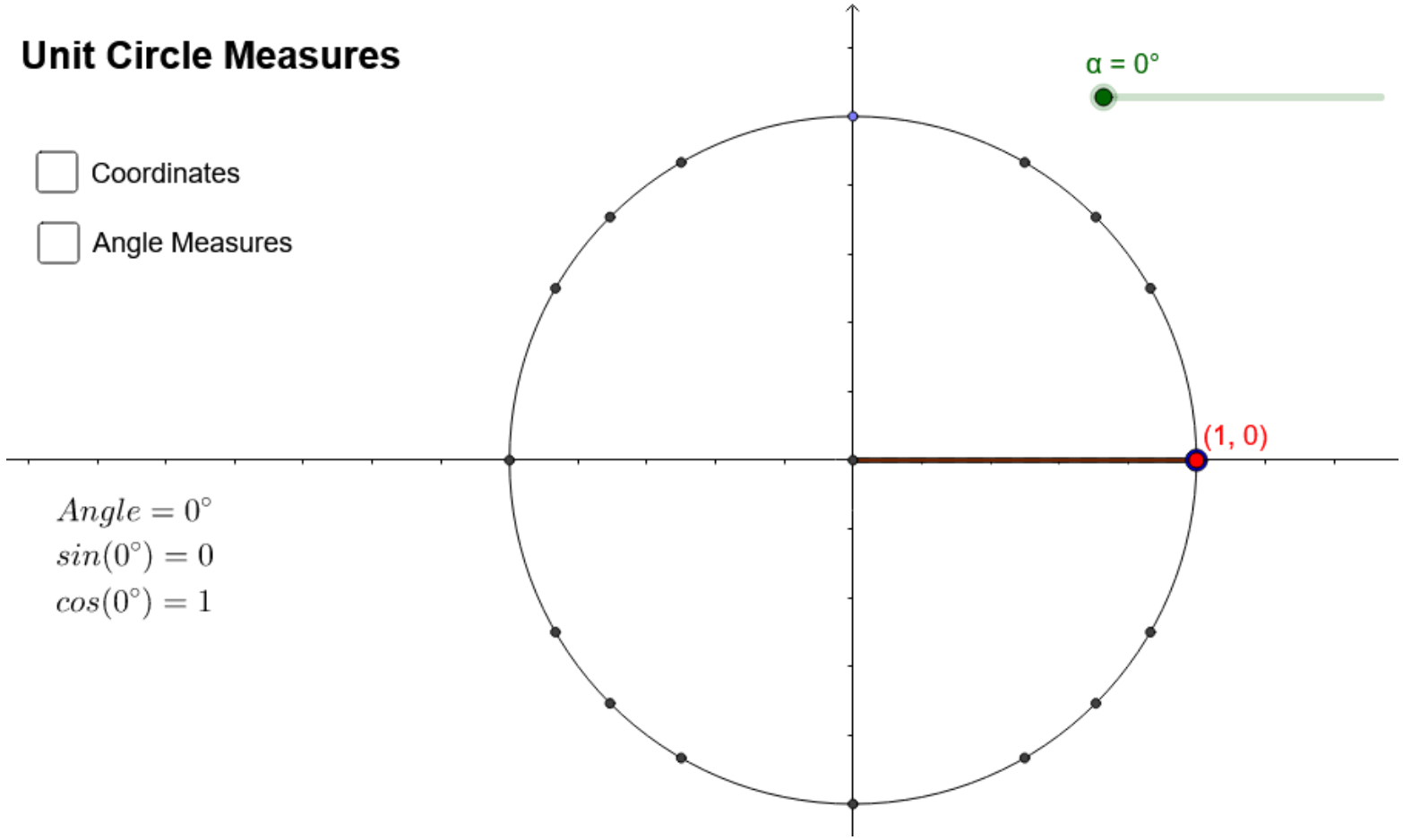
- Halbierung eines Vektors

$$0{,}5 \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{x}{2} \\ \frac{y}{2} \end{pmatrix} \qquad 0.5 \cdot \begin{pmatrix} 2 \\ 5 \end{pmatrix} = \begin{pmatrix} 1 \\ 2.5 \end{pmatrix}$$



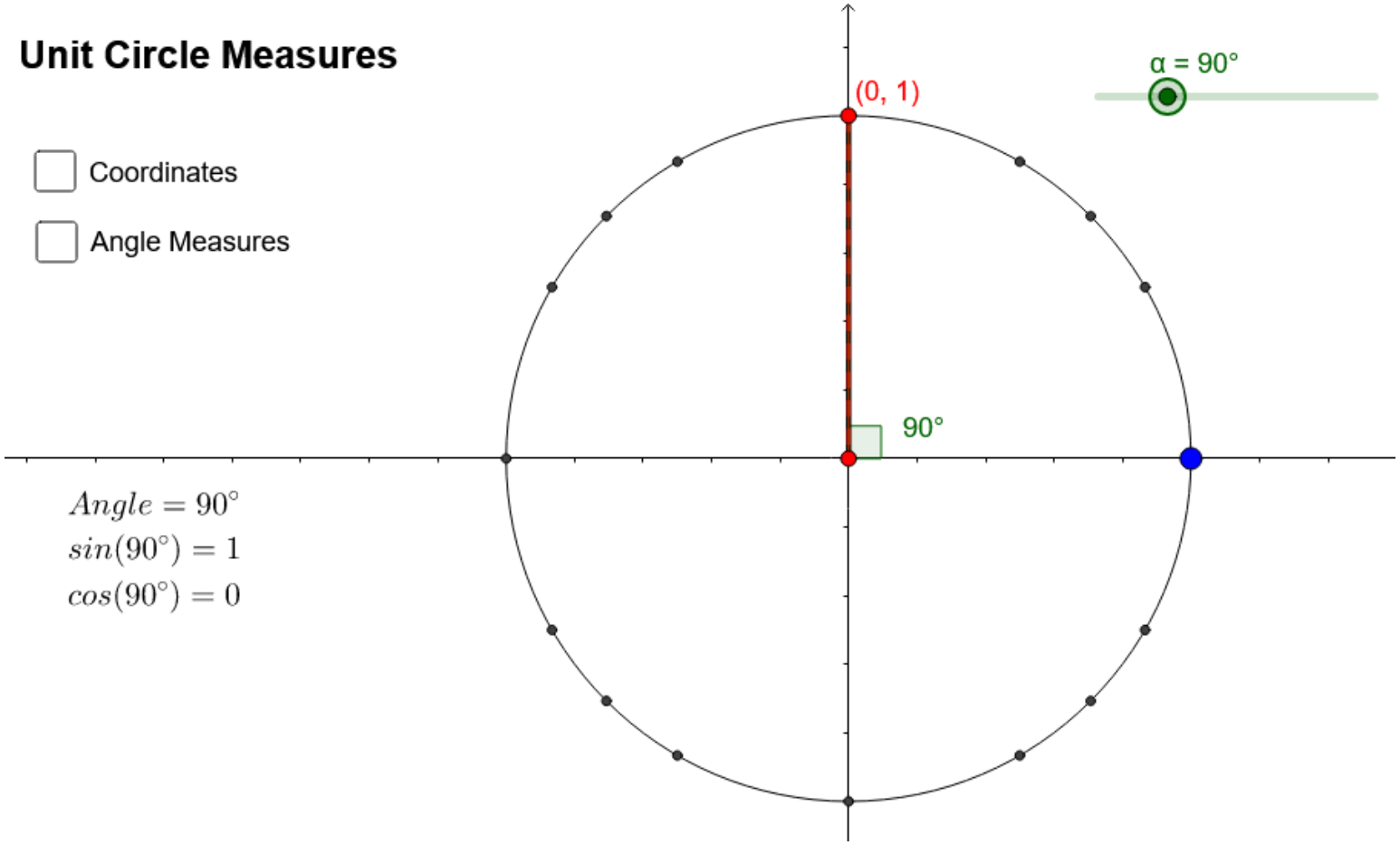
Vektoren – Rotation 2D

Einheitskreis und Winkelfunktionen



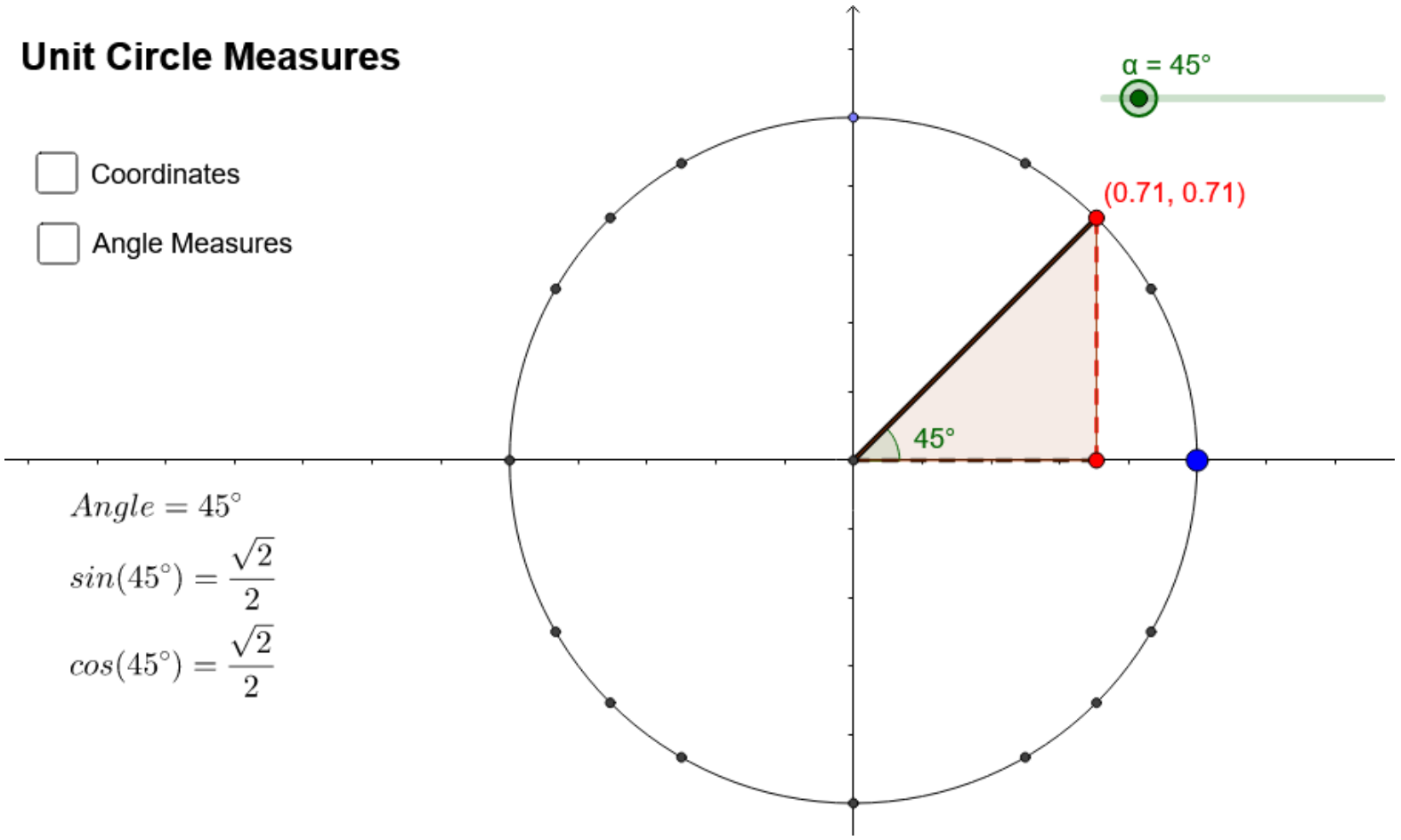
Vektoren – Rotation 2D

Einheitskreis und Winkelfunktionen



Vektoren – Rotation 2D

Einheitskreis und Winkelfunktionen



Vektoren – Rotation 2D

- Rotation gegen den Uhrzeigersinn (Blick in Richtung Ursprung) um den Winkel α

$$x' = x \cdot \cos(\alpha) - y \cdot \sin(\alpha)$$

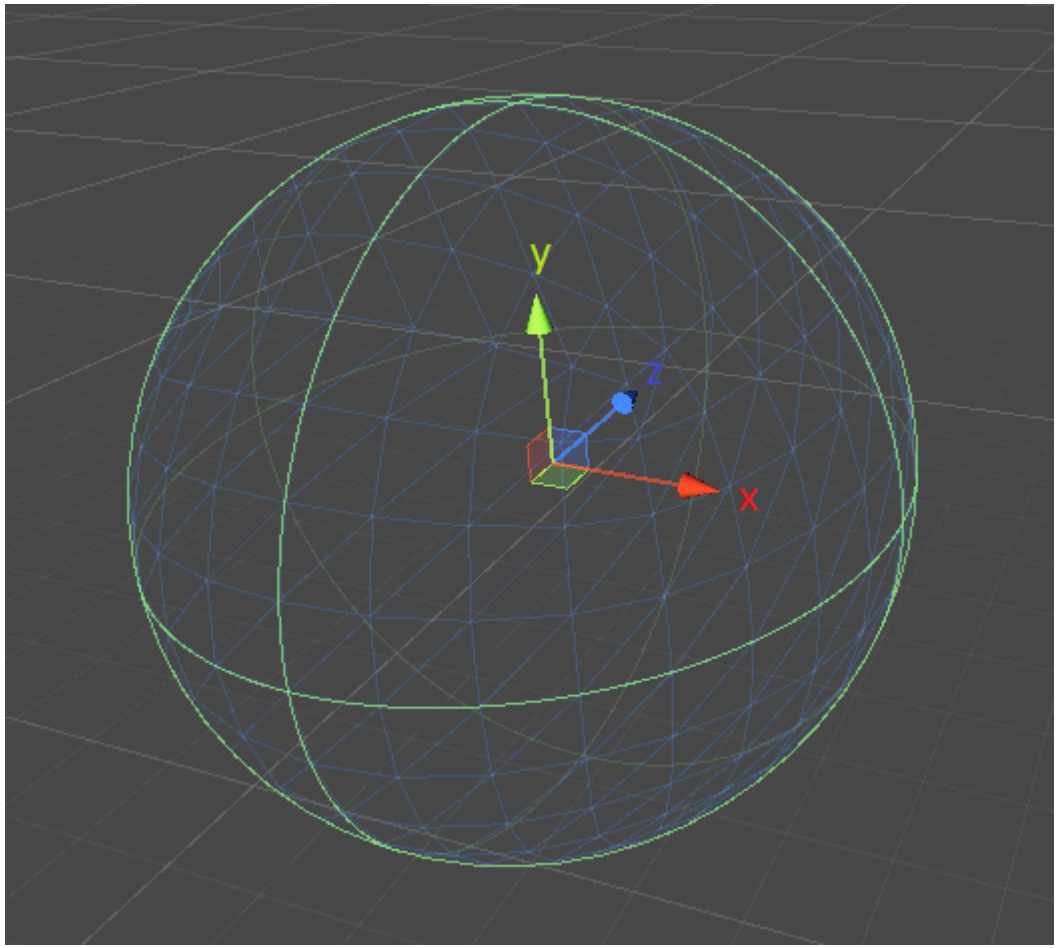
$$y' = x \cdot \sin(\alpha) + y \cdot \cos(\alpha)$$

Vektoren – Rotation 3D

- 3D-Raum hat unendlich viele Ebenen auf denen die Rotation stattfinden kann

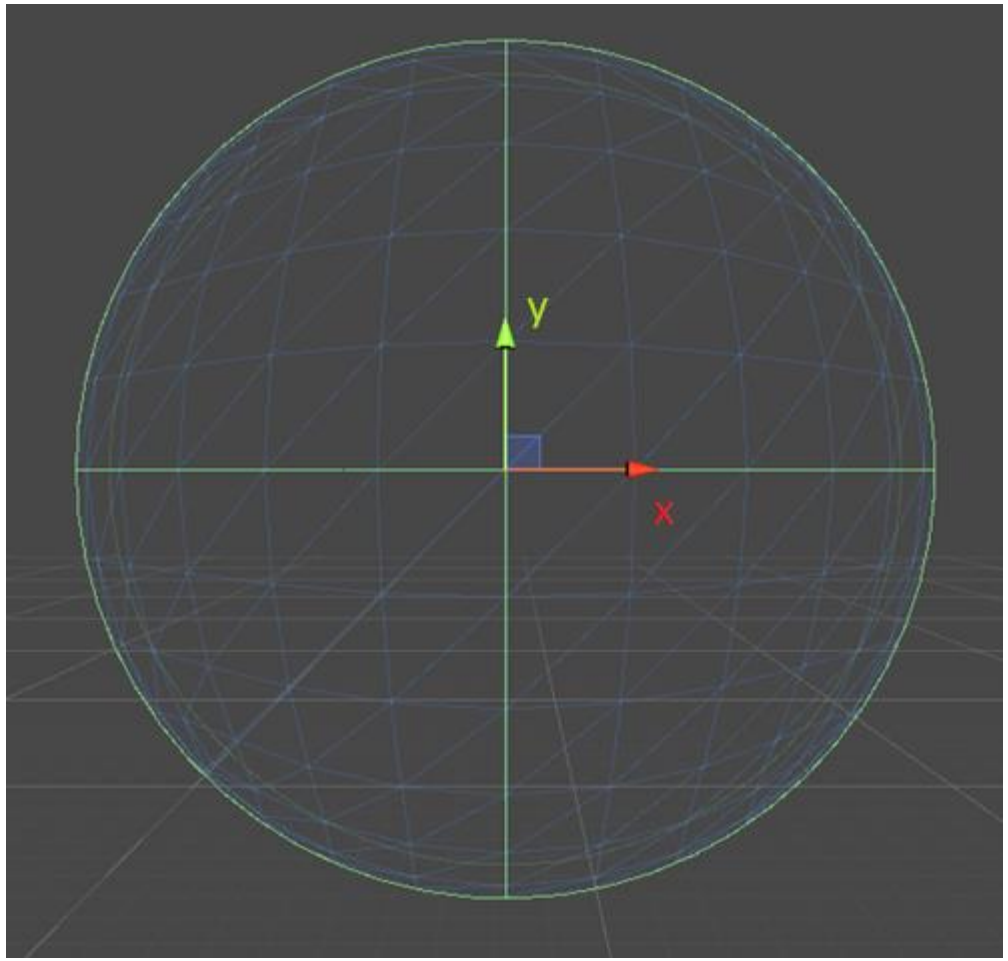
- 3 Ebenen werden durch die Achsen als Normale definiert

- Eine Normale ist ein Vektor der senkrecht auf einer Ebene steht

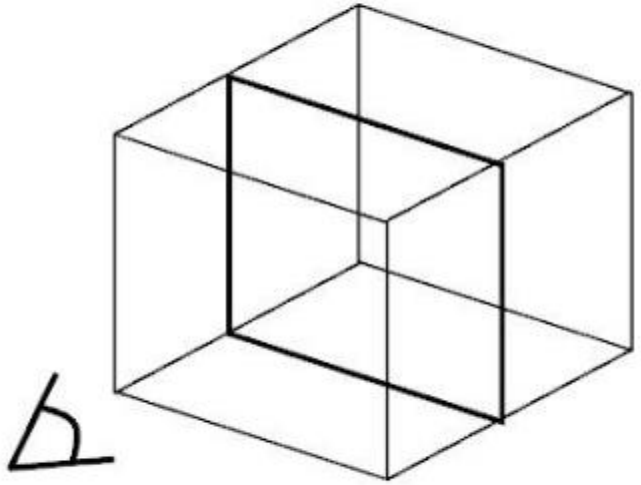


Vektoren - Rotation 3D – um Z-Achse

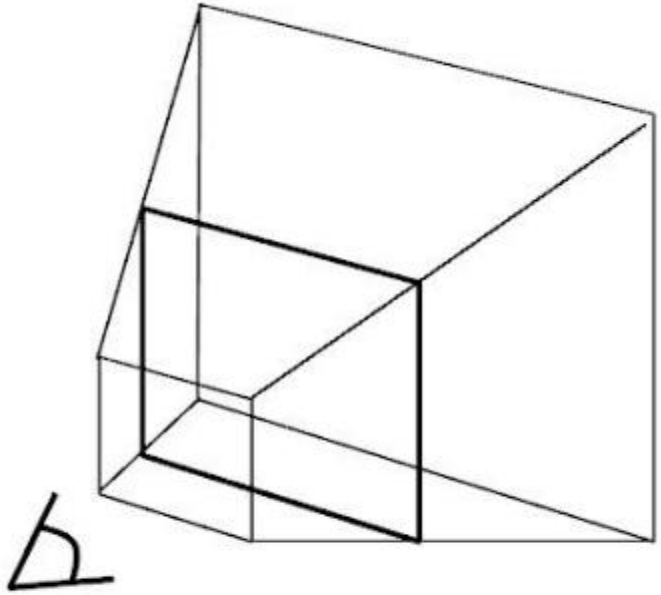
- Blickt man orthografisch entlang der Z-Achse…
- …gilt wieder die Rotation im Einheitskreis



Projektionen



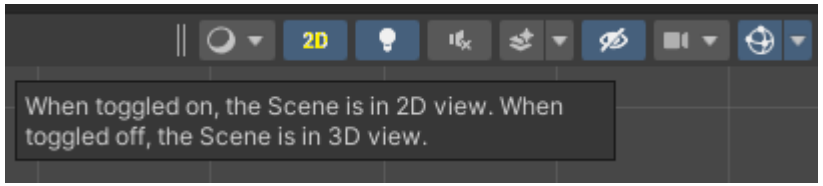
orthographische



perspektivische

Projektionen

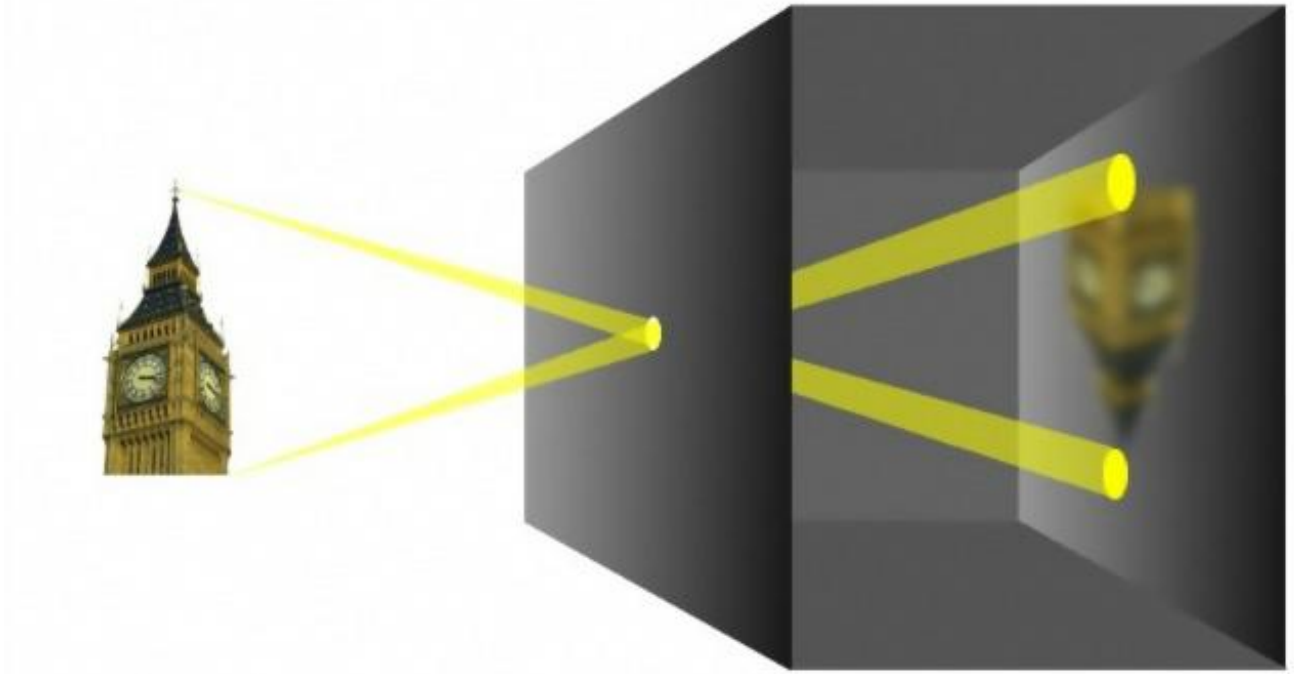
- Orthografische Projektion

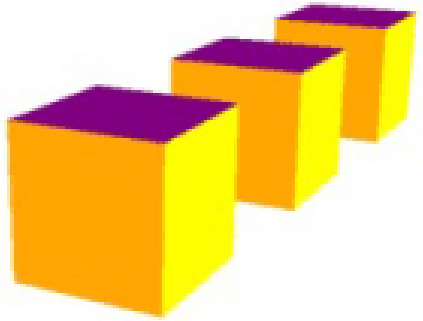


  - Alle Sichtstrahlen laufen parallel
  - Es gibt keinen perspektivischen Eindruck
  - Entfernungen können nicht wahrgenommen werden

Projektionen

- Perspektivische Projektion
  - Verzerrung durch Konzentration aller Sehstrahlen in einem Punkt
  - Weiter entfernte Objekte erscheinen Kleiner als weiter vorne stehende





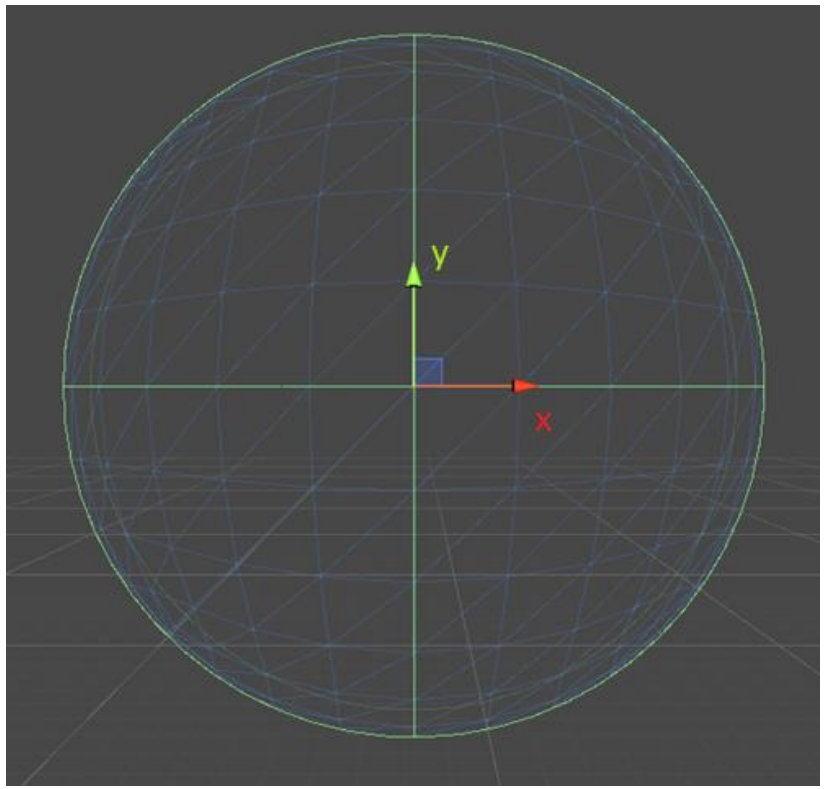
Vektoren - Rotation 3D – um Z-Achse

$$x' = x \cdot \cos(\alpha) - y \cdot \sin(\alpha)$$

$$y' = x \cdot \sin(\alpha) + y \cdot \cos(\alpha)$$

$$z' = z$$

- Die Koordinate um dessen Achse rotiert wird verändert sich nicht



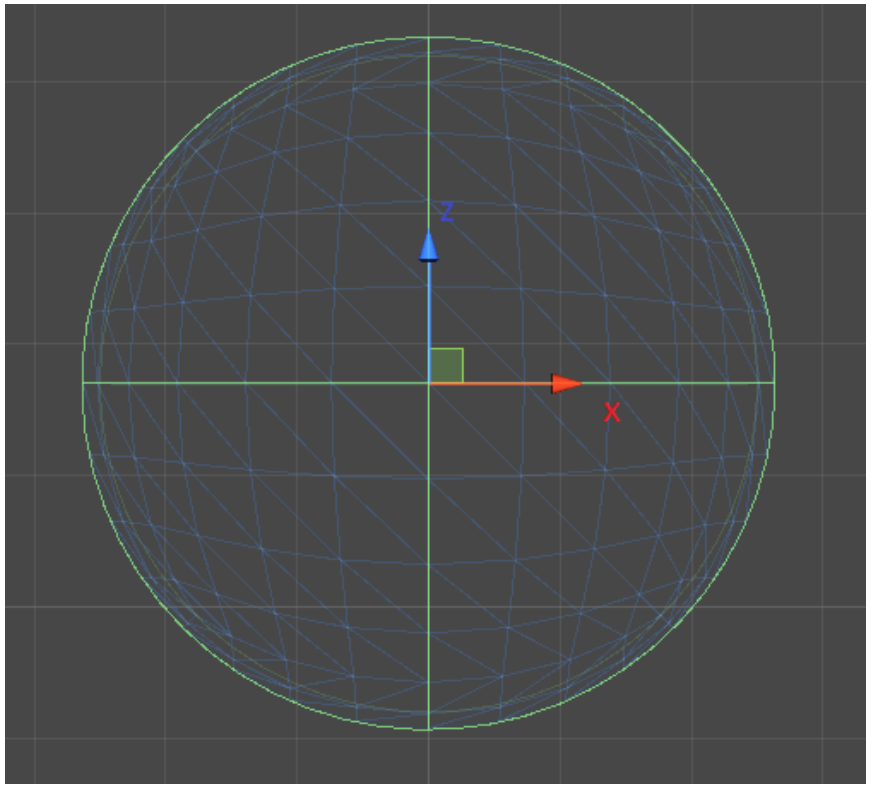
Vektoren - Rotation 3D – um Y-Achse

$$x' = x \cdot \cos(\alpha) + z \cdot \sin(\alpha)$$

$$y' = y$$

$$z' = -x \cdot \sin(\alpha) + z \cdot \cos(\alpha)$$

- Die Koordinate um dessen Achse rotiert wird verändert sich nicht



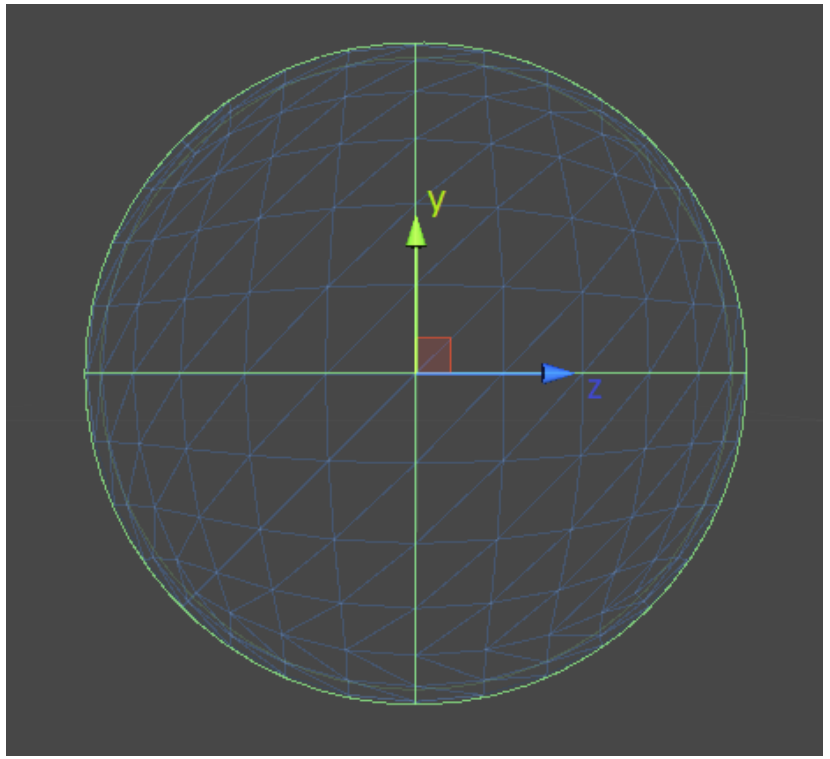
Vektoren - Rotation 3D – um X-Achse

$$x' = \mathrm{x}$$

$$y' = y \cdot \cos(\alpha) - z \cdot \sin(\alpha)$$

$$z' = y \cdot \sin(\alpha) + z \cdot \cos(\alpha)$$

- Die Koordinate um dessen Achse rotiert wird verändert sich nicht



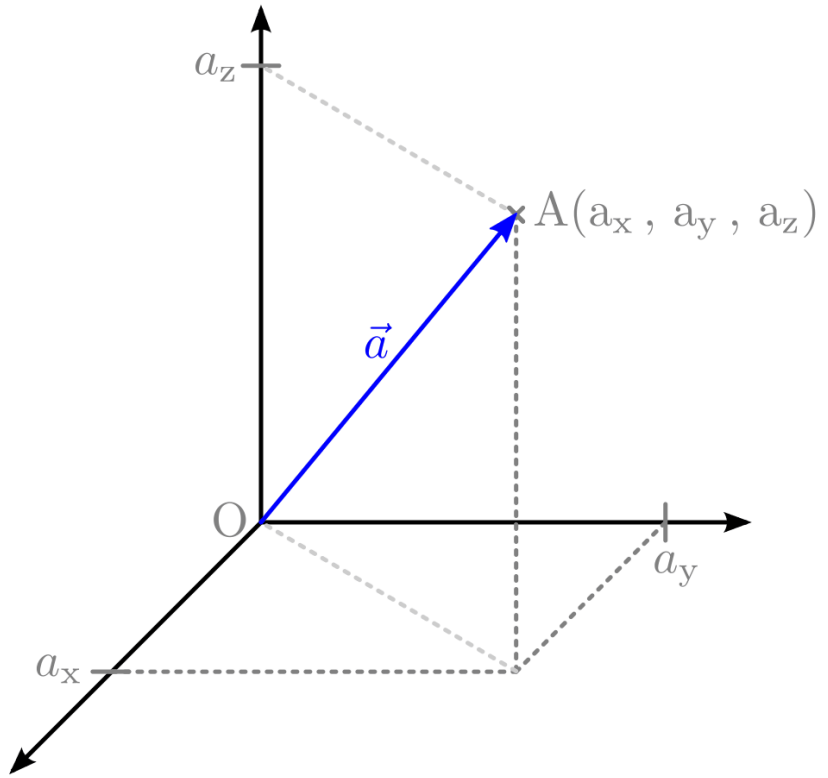
Transformation

- unter Transformation fällt:
  - Translation
  - Rotation
  - Skalierung
- Unity verwaltet Position, Rotation und Skalierung in der Property `transform` jedes Game-Objekts.
- https://docs.unity3d.com/ScriptReference/Transform.html

Vektoren - Länge

- Die Länge eines Vektors lässt sich mit dem Satz von Pythagoras berechnen:

$$l = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}$$

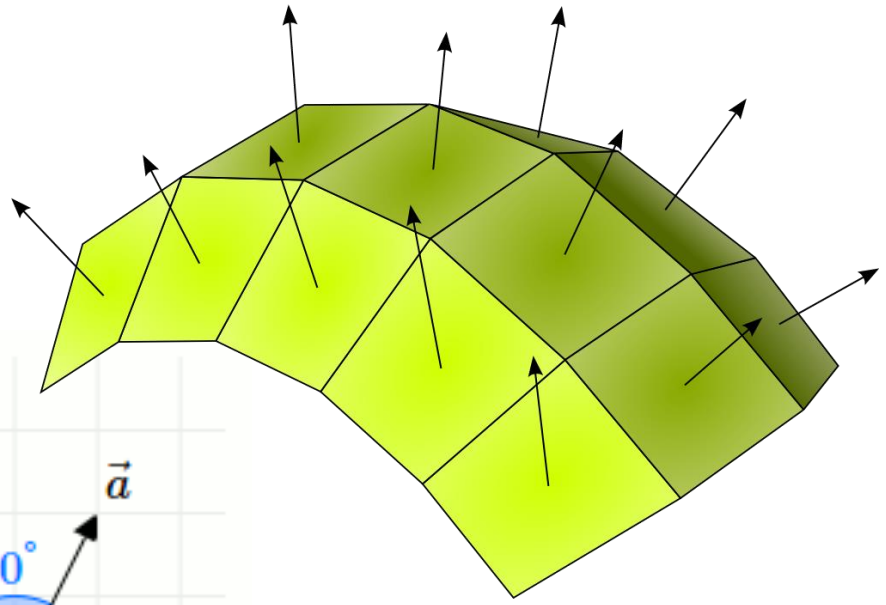


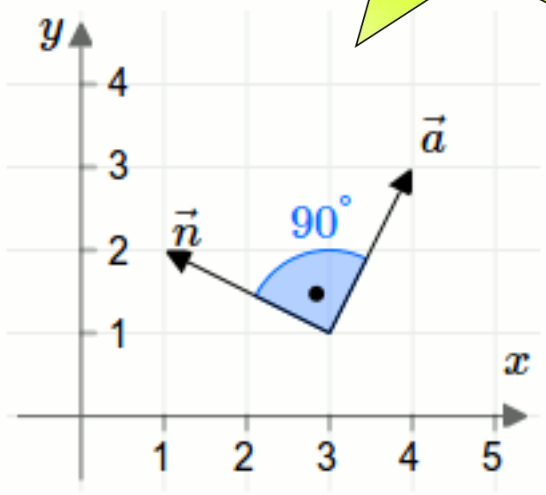
Normalvektor

- Ein Vektor ist ein Normalvektor bezüglich eines anderen Vektors, wenn **die jeweiligen Richtungen der Vektoren zueinander um 90° gedreht sind**.
- Das Skalarprodukt eines Vektors und eines dazugehörigen Normalvektors ist gleich Null.

$$a = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \qquad n = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} -2 \\ 1 \end{pmatrix} = 1 \cdot -2 + 2 \cdot 1 = 0$$





Coding Task

- clone Repository - git@git.htl-hl.ac.at:HOEF/ITP4-HOEF6-2223.git

- erstellt einen git-Branch auf Basis des main-Branchs für euer Team: gruppeA/gruppeB

Coding Task

- **vom master-Branch mergen**

1. `git fetch`
   um Änderungen am master-Branch und sämtlichen anderen Remote-Branches zu erkennen

2. `git checkout gruppe<X>`
   stelle sicher auf deinem Team-Branch zu sein

3. `git merge origin/master`
   Änderungen vom Master-Branch in den Team-Branch mergen

Game-Engine … a definition

- Eine Game-Engine ist definiert als Sammlung von Software-Tools oder APIs, gebaut um Spieleentwicklung zu optimieren.

- Typischerweise stellt eine Game-Engine zumindest folgende Funktionalität zur Verfügung:

1. Game Loop

2. 2D und/oder 3D Rendering-Engine
   grafische Darstellung von Objekten und Bildern durch Ansprechen der Grafikeinheit + Speichermanagement

Game-Engine

- meist beinhalten Game-Engines wie Unity aber wesentlich mehr Komponenten:

3. Physiksimulation und Kollisionserkennung
4. Animationen
5. Audio
6. Multiplayer und Netzwerktools
7. (World) Editor
8. Scripting
9. VR Tools
10. etc...

Game Engines

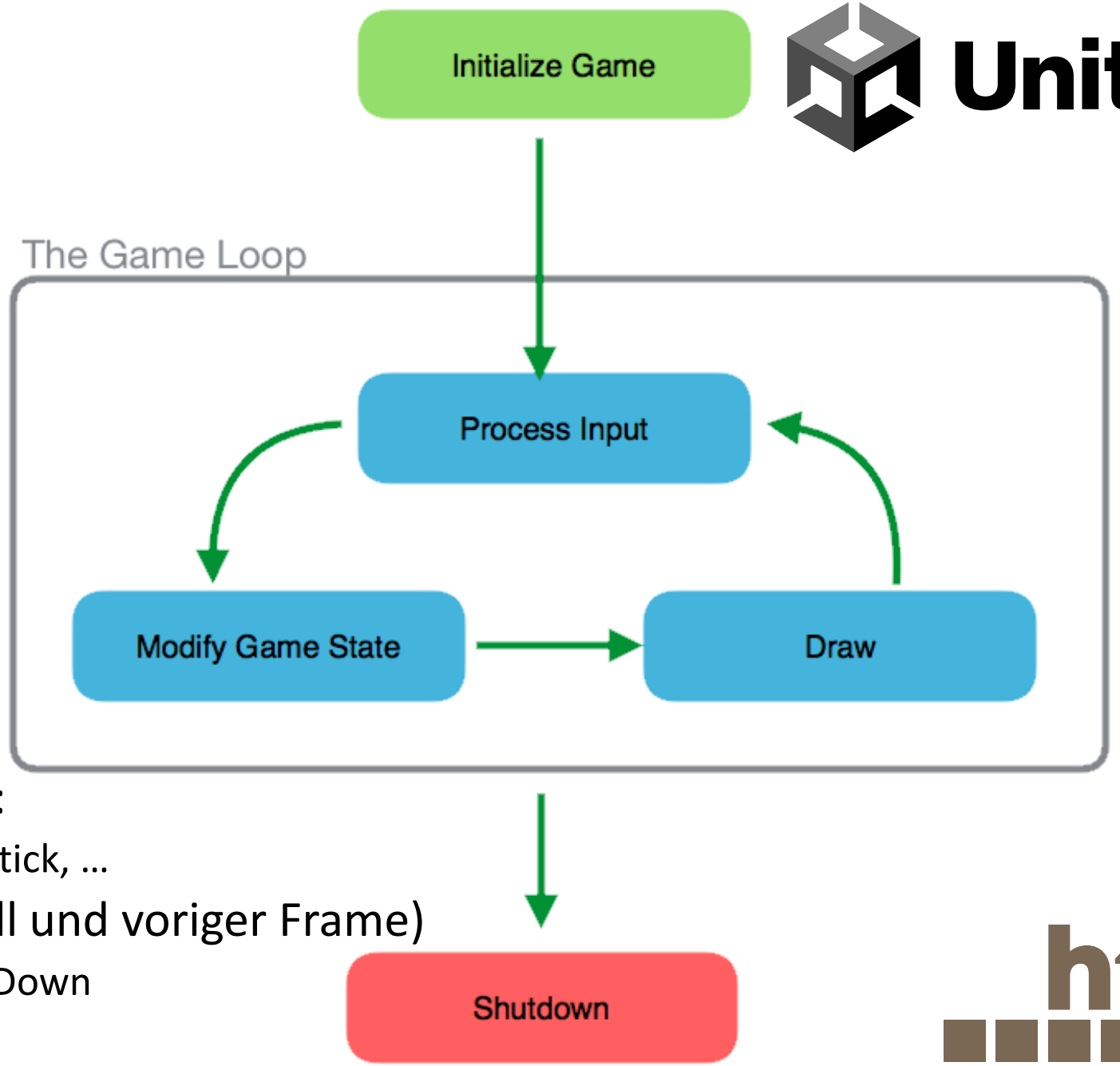
- aktuelle Beispiele für Game-Engines:
  - Unity - https://unity.com/de
  - Unreal (by Epic Games) - https://www.unrealengine.com/en-US
  - Godot - https://godotengine.org/
  - CryEngine - https://www.cryengine.com/
  - GameMaker: Studio (Point-and-Click) - https://gamemaker.io/de/gamemaker

- keine Game-Engines:
  - DirectX, OpenGL, Vulkan, Three.js, …

Game Loop

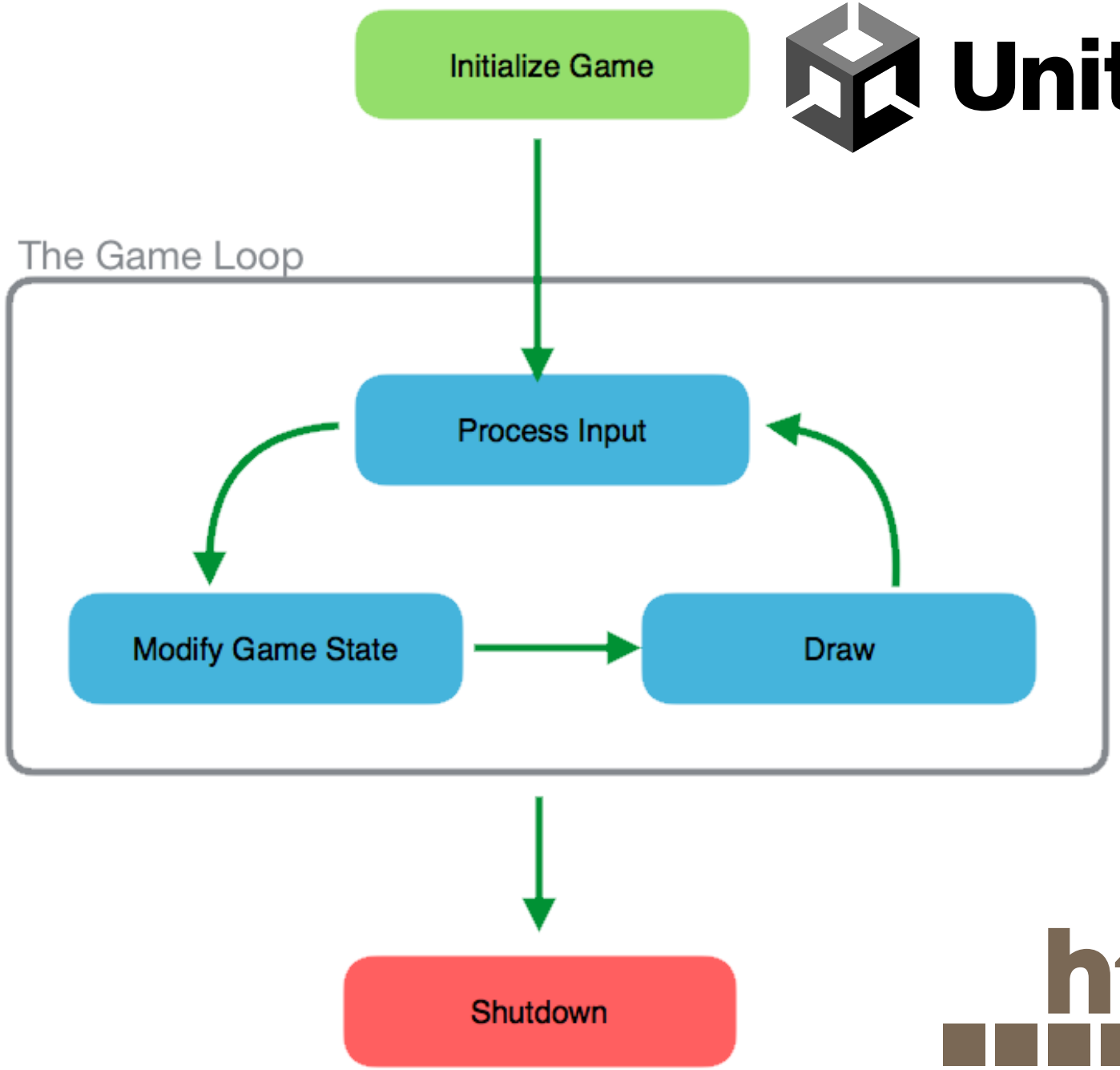
- Initialize:
  - starte Logging
  - erstelle Window
  - lade Assets

- Verarbeite Input:
  - Einlesen des Zustandes aller relevanten Eingabegeräte:
    - Keyboard, Maus, Controller, Joystick, …
  - Speichern der Zustände (aktuell und voriger Frame)
    - Dadurch können KeyUp und KeyDown Events erkannt werden



Game Loop

- Update (Modify):
  - Gamelogik ausführen
  - Transformationen durchführen
  - Physik, Netzwerk, AI, …
- Draw:
  - leere den Screen
  - zeichne alle Game-Objekte, die sichtbar sind am Screen
- Shutdown:
  - Freigabe aller Ressourcen,
  - Beendigung Game Prozess



Game Loop Timing

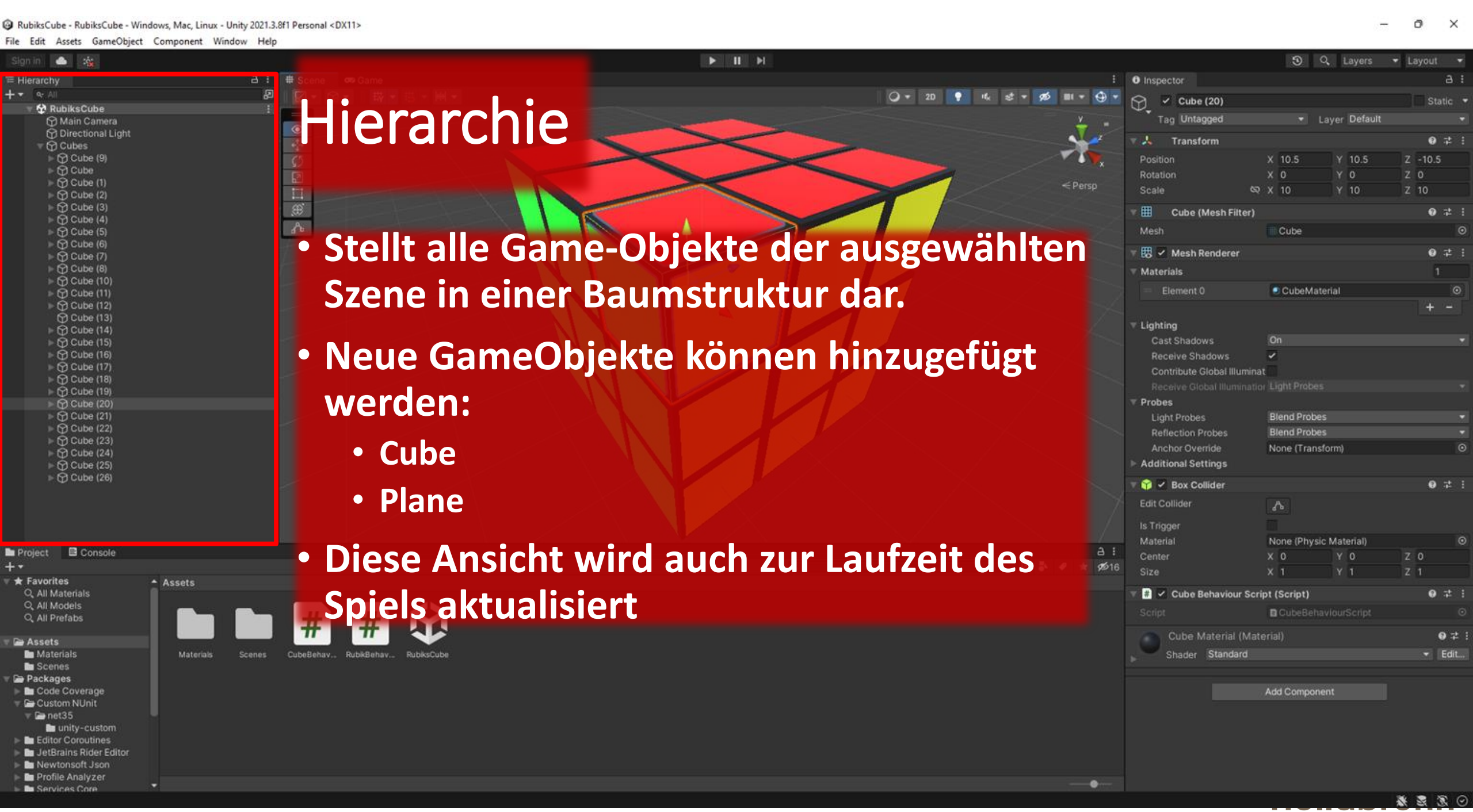
- bei Game-Objekte mit Skript-Komponente findet ein `Update`-Aufruf statt
- `Time.deltaTime` liefert die seit dem letzten Update verstrichene Zeit in Sekunden

- z.B. bei 60 frames per second (fps)
  - Ein Durchlauf der Game Loop darf nicht länger dauern als:

$$\frac{1s}{60} = 16{,}67ms$$

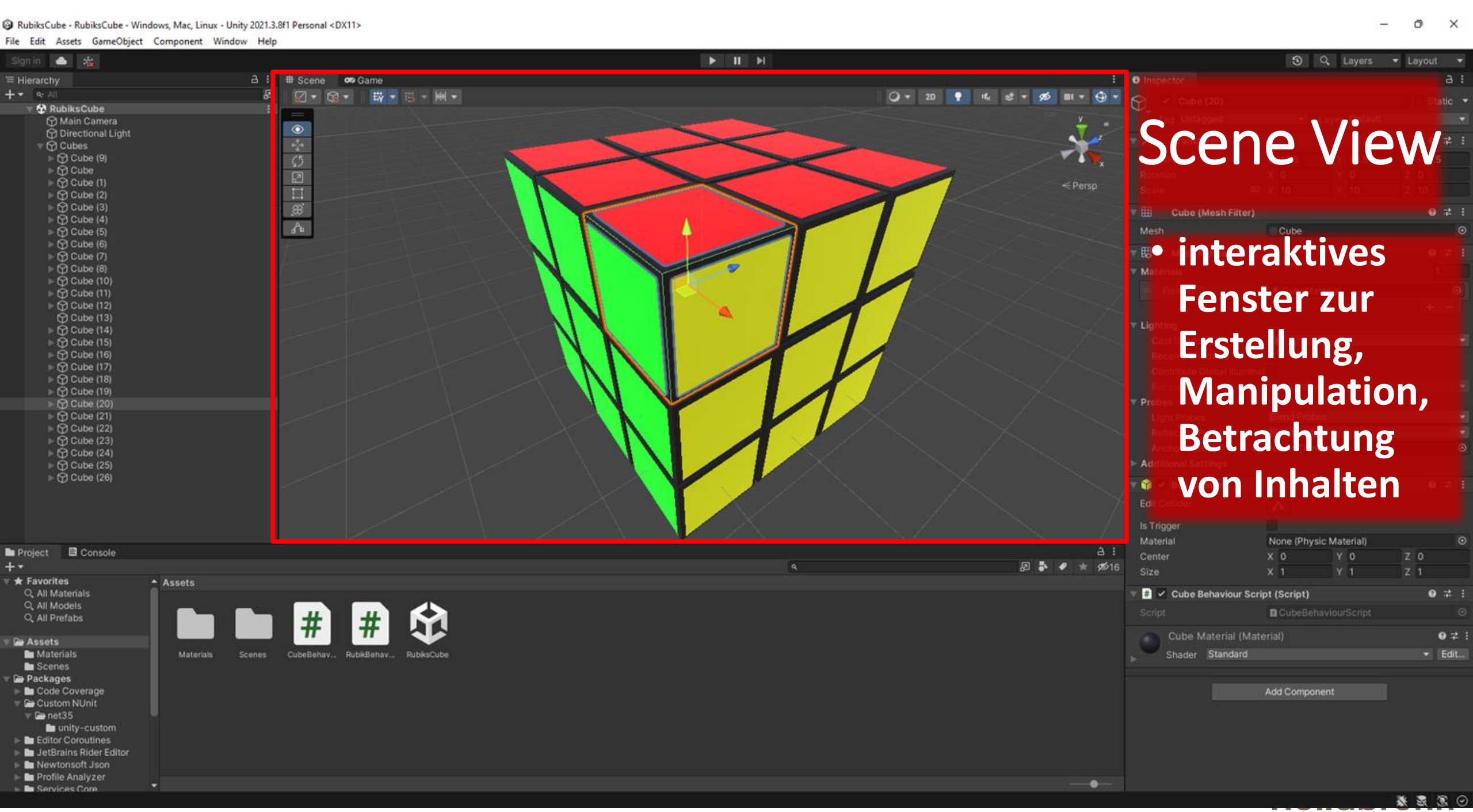
Unity Editor

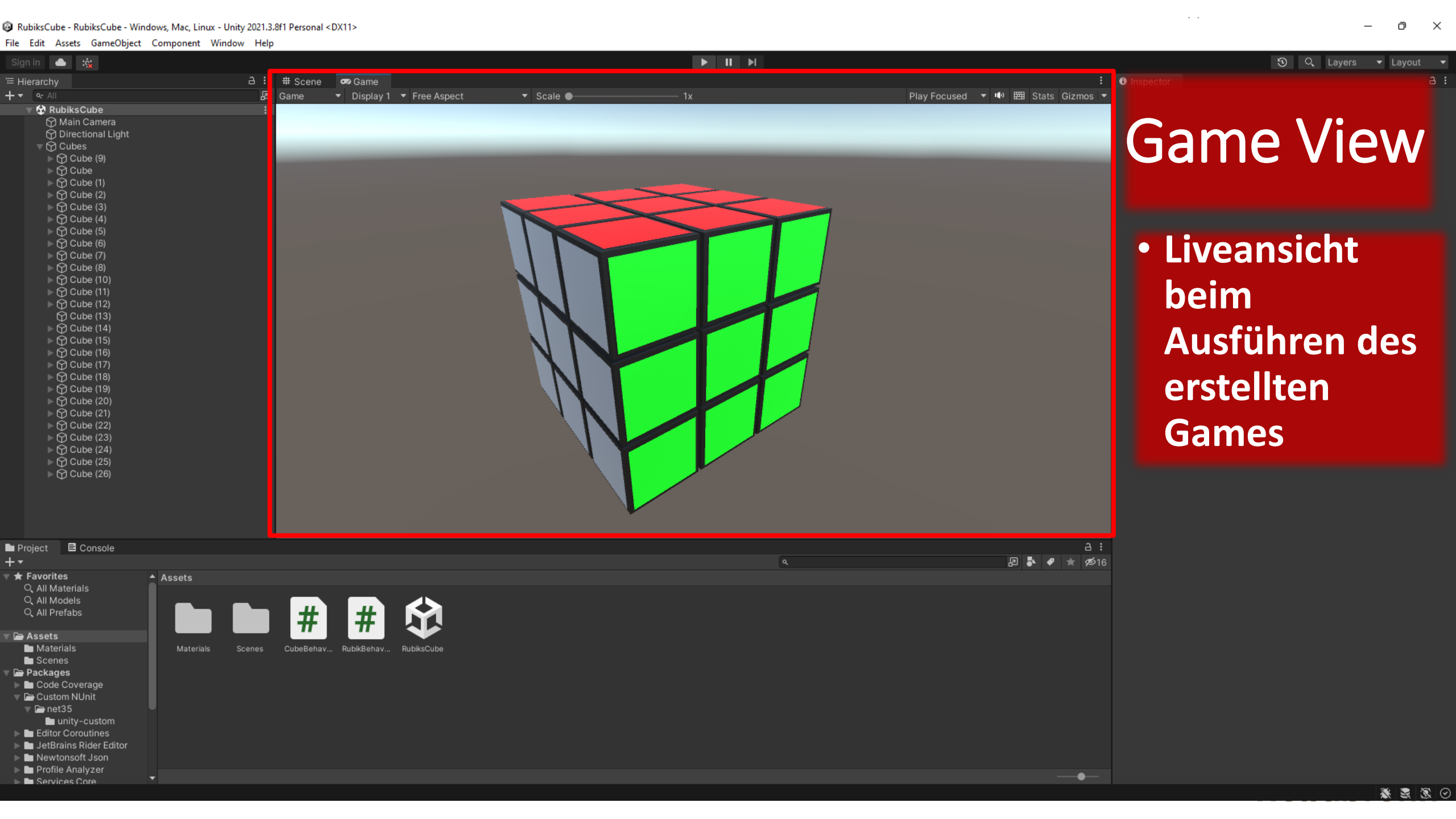
- Scene / Game View
- Hierarchy
- Inspector
- Project / Assets
- Console



Hierarchie

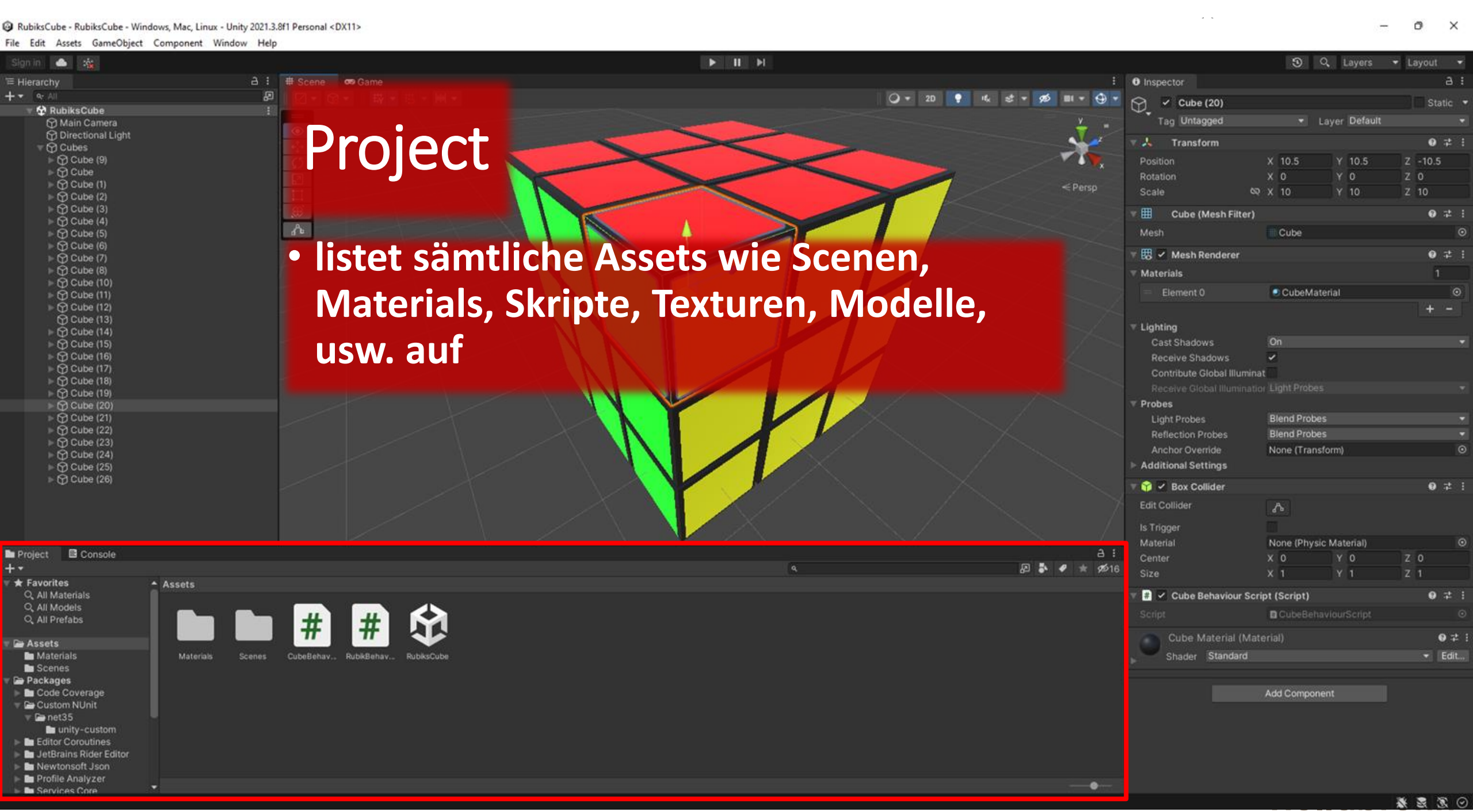
- Stellt alle Game-Objekte der ausgewählten Szene in einer Baumstruktur dar.
- Neue GameObjekte können hinzugefügt werden:
  - Cube
  - Plane
- Diese Ansicht wird auch zur Laufzeit des Spiels aktualisiert


RubiksCube - RubiksCube - Windows, Mac, Linux - Unity 2021.3.8f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
RubiksCube
Main Camera
Directional Light
Cubes
Scene
Game
Persp
Inspector
Cube (Mesh Filter)
Mesh
Cube
Is Trigger
Material
None (Physic Material)
Center
Size
Cube Behaviour Script (Script)
Script
CubeBehaviourScript
Cube Material (Material)
Shader
Standard
Add Component
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Materials
Scenes
Packages
Code Coverage
Custom NUnit
net35
unity-custom
Editor Coroutines
JetBrains Rider Editor
Newtonsoft Json
Profile Analyzer
Services Core
Assets
Materials
Scenes
CubeBehav...
RubikBehav...
RubiksCube
Scene View
• interaktives Fenster zur Erstellung, Manipulation, Betrachtung von Inhalten


RubiksCube - RubiksCube - Windows, Mac, Linux - Unity 2021.3.8f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
RubiksCube
Main Camera
Directional Light
Cubes
Scene
Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos
Inspector
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Materials
Scenes
Packages
Code Coverage
Custom NUnit
net35
unity-custom
Editor Coroutines
JetBrains Rider Editor
Newtonsoft Json
Profile Analyzer
Services Core
Materials
Scenes
CubeBehav...
RubikBehav...
RubiksCube
Game View
• Liveansicht beim Ausführen des erstellten Games

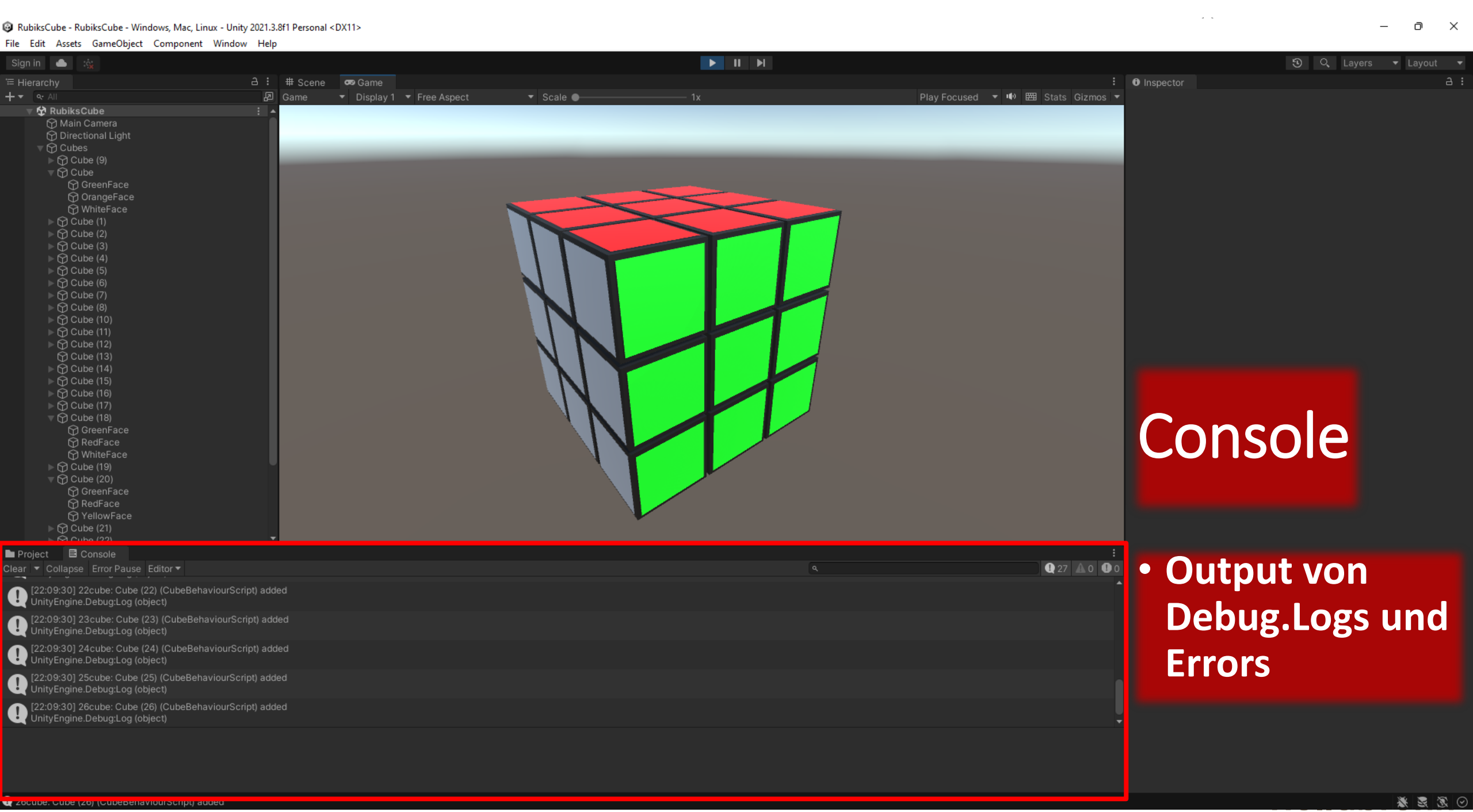
Inspector

- macht Eigenschaften von selektierten Game-Objekten sichtbar und bietet an diese zu verändern
- listet alle Komponenten eines Game-Objekts auf (Transform, Material, Script,...)



Project

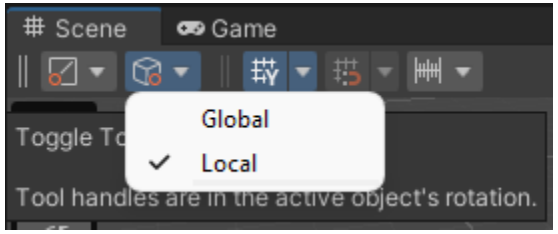
- listet sämtliche Assets wie Scenen, Materials, Skripte, Texturen, Modelle, usw. auf


RubiksCube - RubiksCube - Windows, Mac, Linux - Unity 2021.3.8f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
RubiksCube
Main Camera
Directional Light
Cubes
Cube (9)
Cube
GreenFace
OrangeFace
WhiteFace
Cube (1)
Cube (2)
Cube (3)
Cube (4)
Cube (5)
Cube (6)
Cube (7)
Cube (8)
Cube (10)
Cube (11)
Cube (12)
Cube (13)
Cube (14)
Cube (15)
Cube (16)
Cube (17)
Cube (18)
GreenFace
RedFace
WhiteFace
Cube (19)
Cube (20)
GreenFace
RedFace
YellowFace
Cube (21)
Scene
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos
Inspector
Project
Console
Clear
Collapse
Error Pause
Editor
[22:09:30] 22cube: Cube (22) (CubeBehaviourScript) added
UnityEngine.Debug:Log (object)
[22:09:30] 23cube: Cube (23) (CubeBehaviourScript) added
UnityEngine.Debug:Log (object)
[22:09:30] 24cube: Cube (24) (CubeBehaviourScript) added
UnityEngine.Debug:Log (object)
[22:09:30] 25cube: Cube (25) (CubeBehaviourScript) added
UnityEngine.Debug:Log (object)
[22:09:30] 26cube: Cube (26) (CubeBehaviourScript) added
UnityEngine.Debug:Log (object)
Console
• Output von Debug.Logs und Errors

Explore the Unity Editor - Tutorial

https://learn.unity.com/tutorial/explore-the-unity-editor-1#

Spaces



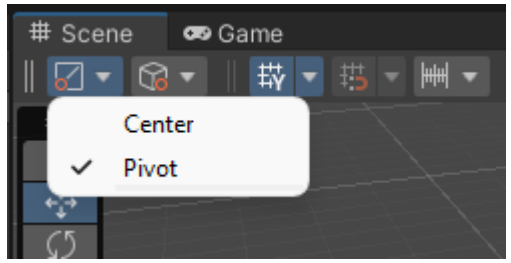
- Local Space
  Position eines Objekts in Relation zu einem anderen Objekt z.B. Parent (relativ)

- World(Global) Space
  Position eines Objekts im gesamten Raum der Unity Szene (absolut)

- Legt man ein Child-Object an sind, die Positionen des Parents und des Childs im World Space identisch.
  Der Local Space des Child-Objects ist initial bei gleicher Position

```
X: 0   Y: 0   Z: 0
```

Pivot & Center



- bei Game-Objekten kann zwischen Pivot und Center unterschieden werden:

  - Pivot (Dreh- und Angelpunkt):
    ist abhängig von der geometrischen Figur ein definierter Anker-Punkt (muss nicht der Mittelpunkt sein)

  - Center:
    der Mittelpunkt (wird aufgrund der Grenzen eines Objekts berechnet)

$$\frac{Summe\ aller\ Vertices}{Anzahl\ aller\ Vertices}$$

Unity meets GIT

- Probleme

  - Temporäre Dateien
    Durch das Öffnen und Schließen des Editors werden einige dieser Dateien erstellt, geändert oder gelöscht

  - Objekt-Referenzen
    Unity stellt Beziehungen/Abhängigkeiten mittels zufällig erstellten GUIDs her und verwendet diese in `.meta` Dateien. Gleichzeitige Anlagen/Änderungen führen daher zu Konflikten.

Unity meets GIT

- Probleme

  - Unauflösbare Konflikte
    Abhängig von den Editor-Einstellungen werden viele/alle Dateien im Binär-Format gespeichert. Änderungen von zumindest 2 Entwicklern in der selben binären Datei können nicht manuell gelöst werden.

  - Große Dateien
    Assets für 3D-Models, Sounds, Texturen usw. können durch ihre Dateigröße die Performance von GIT beeinflussen und Speicher am GIT-Server verschwenden.

Unity meets GIT

- Lösungen
  - Unity-Spezifisches `.gitignore`
    https://github.com/github/gitignore/blob/main/Unity.gitignore
  - ** Wildcard hinzufügen, sodass alle Ebenen berücksichtigt werden
  - zusätzlich noch je nach Entwicklungsumgebung ignore-Definitions (Rider, Visual Studio, VS Code, …)
  - Unity konfigurieren
    - `Edit > Project Settings > Editor`
    - `Version Control / Mode: „Visible Meta Files“`
    - `Editor > Asset Serialization / Mode: „Force Text“`
    - `File > Save Project`
  - Git Large File Storage (Git LFS) verwenden (falls vorhanden)

Generate C# Solution

- Unity konfigurieren
  - `Edit > Preferences > External Tools`
  - `External Script Editor > z.B. Rider`
  - `Check:`
    `Embedded packages`
    `Local packages`
    `Registry packages`
    `Built-In packages`
    `Packages from unknown sources`

C# Commands

- Input:

```
Input.GetKeyUp(KeyCode.X);
```

- Show Script-Properties in Inspector:

```
[field: SerializeField]
public Vector3 angleToRotate { get; set; }
```

C# Commands

- Find GameObjects:

```
GameObject.Find(“ParentGameObject”);
```

- Get Childs

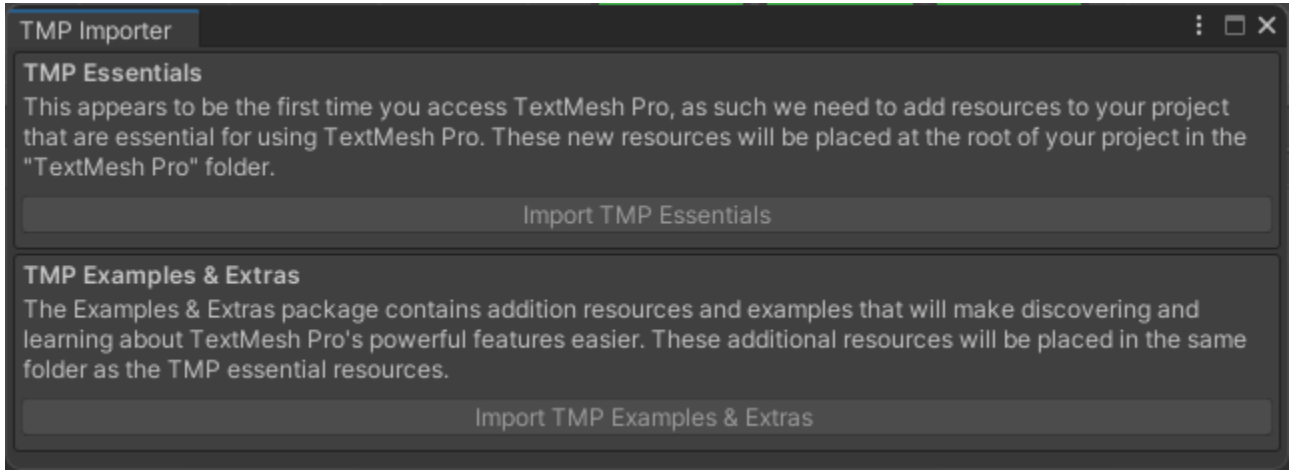
```
parent.transform.childCount;

parent.transform.getChild(i);
```

- Rotate:

```
child.RotateAround(parent.transform.position,
Vector3.up, angle)
```

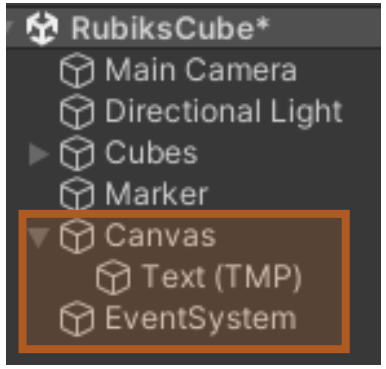
UI Elemente

- Rechtsklick auf das Scene-Object
  - Game Object
  - UI
  - z.B. Text – TextMeshPro

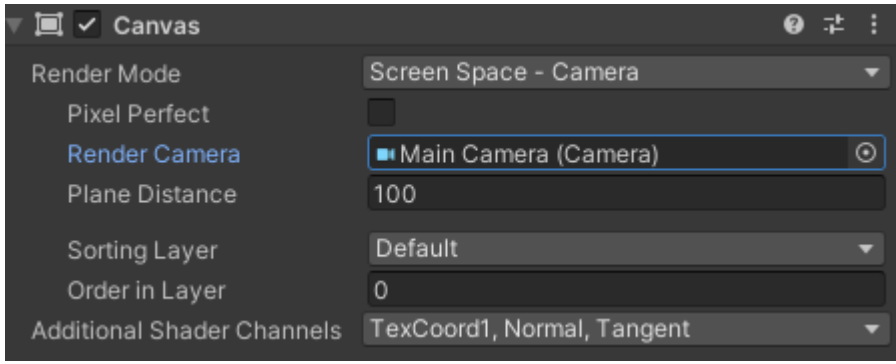


  - Import TMP Essentials + TMP Examples & Extras

  - Erstellt folgende Game Objects:



UI Canvas



- Object: Canvas - Render Mode:
  - ScreenSpace – Overlay

    UI-Elemente platziert am Screen auf oberster Ebene der Szene.
    Bei Änderungen der Auflösung passt sich das Canvas automatisch an.
  - **ScreenSpace – Camera**

    Ähnlich wie Overlay, jedoch wird das UI perspektivisch gerendert.
    Bewirkt ebenfalls Anpassung bei Veränderung der Auflösung.
    Auswahl der Render Camera notwendig!
  - WorldSpace

    Canvas wird behandelt wie ein normales GameObject.
    Wird eingesetzt wenn UI-Elemente teil der Gameworld sein sollen.

C# Commands

- Text ein/ausblenden

```
GameObject.Find("Text (TMP)").GetComponent<TextMeshProUGUI>().text = "";
```

- Farbe eines Materials auslesen

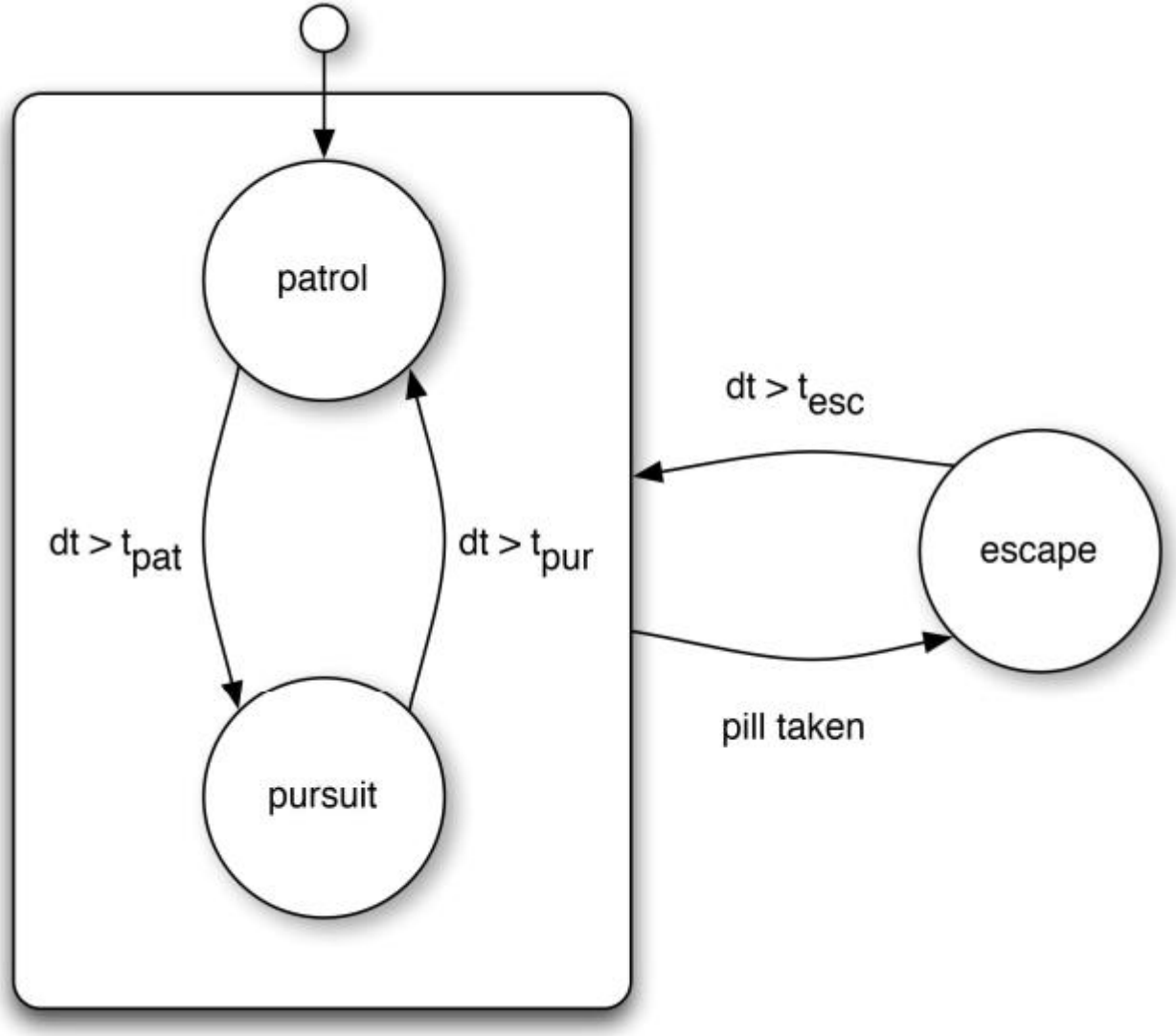
```
face.GetComponent<Renderer>().material.color
```

Coding Tasks

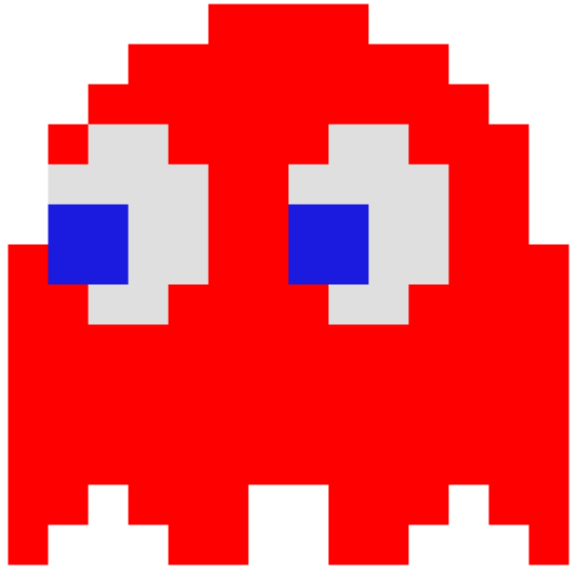
- Rotation der Cubes
  - Inputverarbeitung
  - Animation
  - Einschränkungen der Rotationen (wenn Spalte rotiert, kann Zeile nicht verändert werden)
- Game-Logic Cube solved
- MarkerBehaviour

Pacman AI Spec

Ghosts: General States

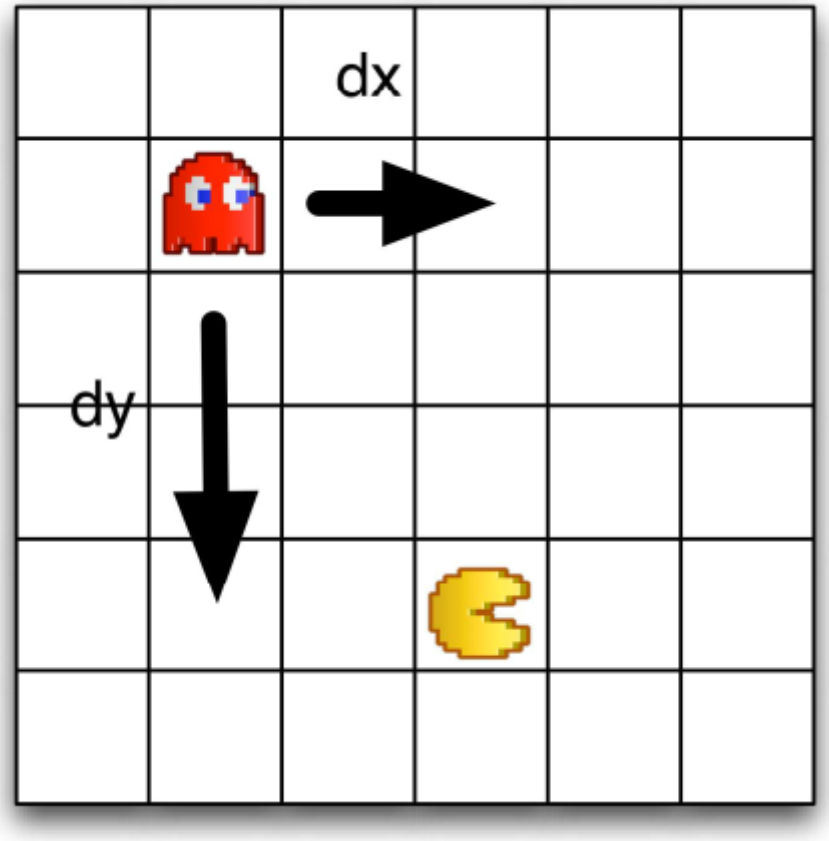


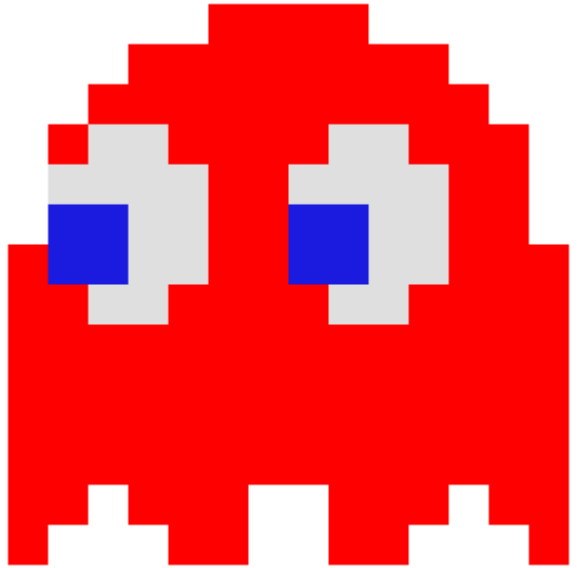
Shadow



- Nickname: Blinky
- As his name implies, Shadow is usually a constant Shadow on Pac-Man's tail. When he's not patrolling the **northeast corner** of the maze, Shadow tries to find the quickest rout to Pac-Man's position.
- When the ghosts are not patrolling their home corners, Blinky will attempt to shorten the distance between Pac-Man and himself.
  - If he has to choose between shortening the horizontal or vertical distance, he will choose to shorten whichever is greatest. For example, if Pac-Man is 4 grid spaces to the left, and 7 grid space above Blinky, Blinky will try to move up before he moves to the left.

Shadow



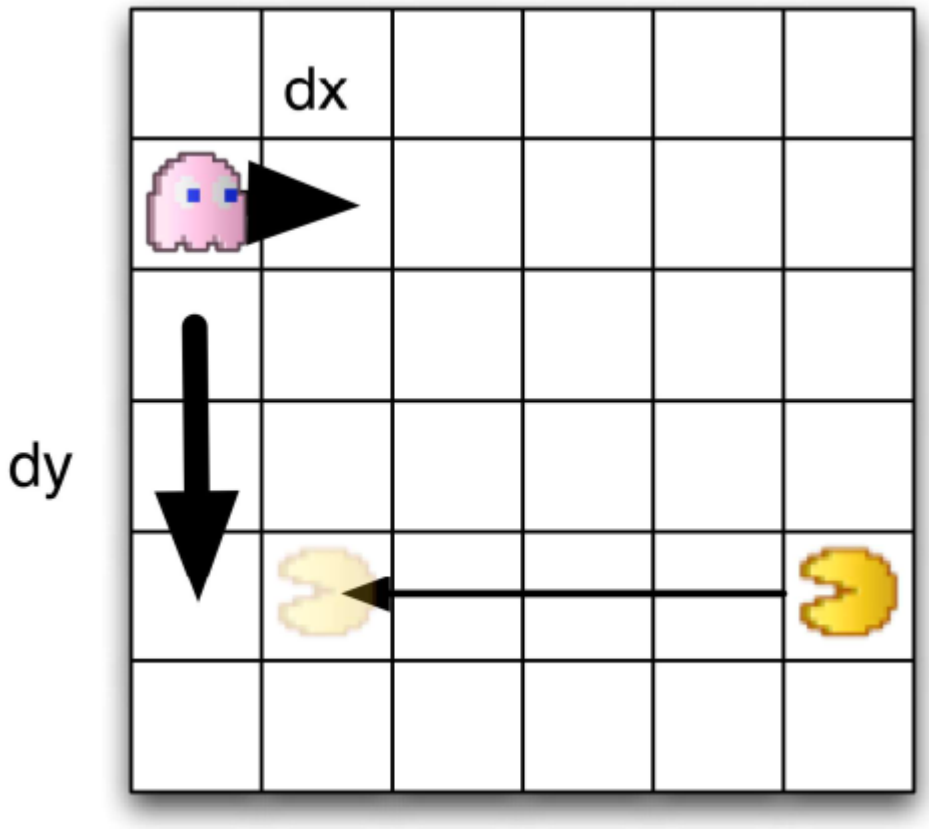


$$\vec{h} = max\left(\begin{pmatrix}\Delta x\\0\end{pmatrix},\begin{pmatrix}0\\\Delta y\end{pmatrix}\right)$$

Speedy

- Nickname: Pinky
- When Speedy isn't patrolling the **northwest corner**, Pinky tries to attack Pac-Man by moving to where he is going to be (that is, a few spaces ahead of Pac-Man's current direction) instead of right where he is, as Blinky does.
- When the ghosts are not patrolling their home corners, Pinky wants to go to the place that is four grid spaces ahead of Pac-Man in the direction that Pac-Man is facing.
  - If Pac-Man is facing up, Pinky wants to go to the location exactly four spaces above Pac-Man. He does this following the same logic that Blinky uses to find Pac-Man's exact location.

Speedy

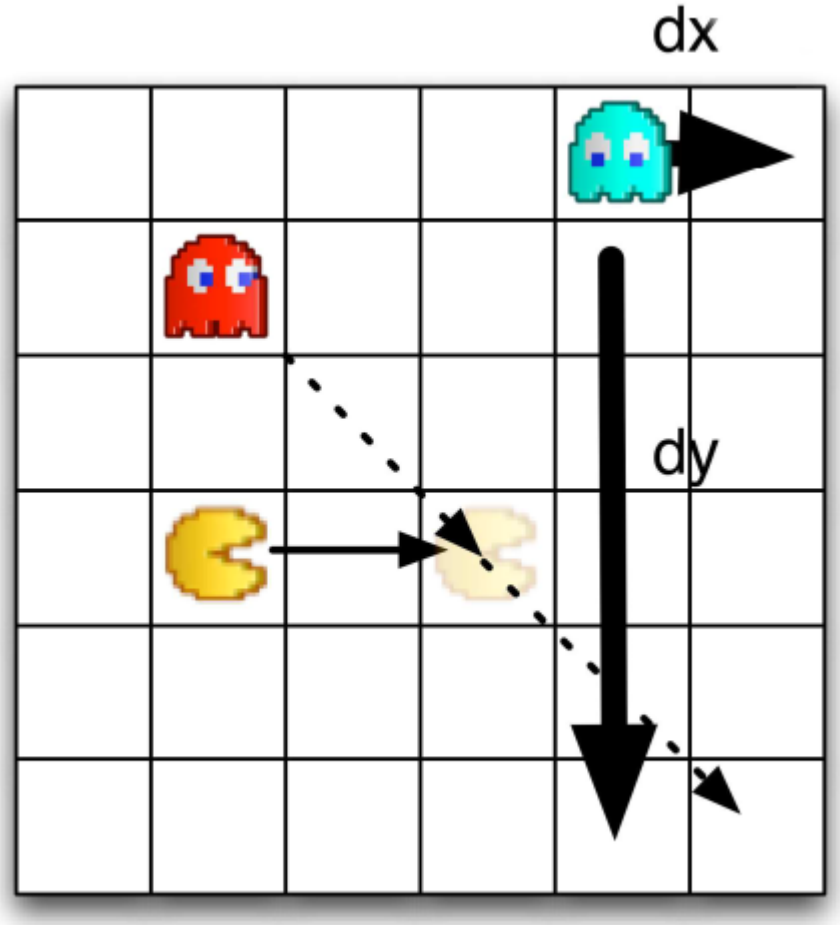


$$\vec{h} = max\left(\begin{pmatrix}\Delta x\\0\end{pmatrix},\begin{pmatrix}0\\\Delta y\end{pmatrix}\right)$$

Bashful

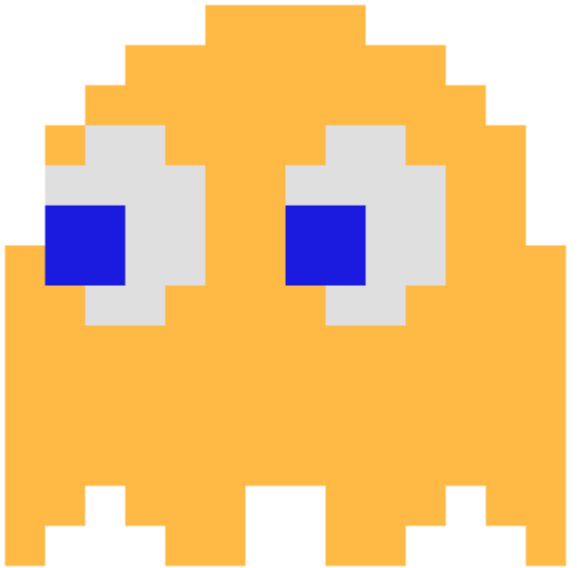
- Nickname: Inky
- Bashful has the most complicated AI of all. When the Ghost is not patrolling its home corner (southeast), Bashful considers two things
  - Shadow's location
  - the location two grid spaces ahead of Pac-Man.
- Bashful draws a line from Shadow to the spot two squares in front of Pac-Man and extends that line twice as far. Therefore, if Bashful is alongside Shadow when they are behind Pac-Man, Bashful will usually follow Shadow the whole time. But if Bashful is in front of Pac-Man when Shadow is behind him, Bashful tends to want to move away from Pac-Man (in reality, to a point very far ahead of Pac-Man).

Speedy



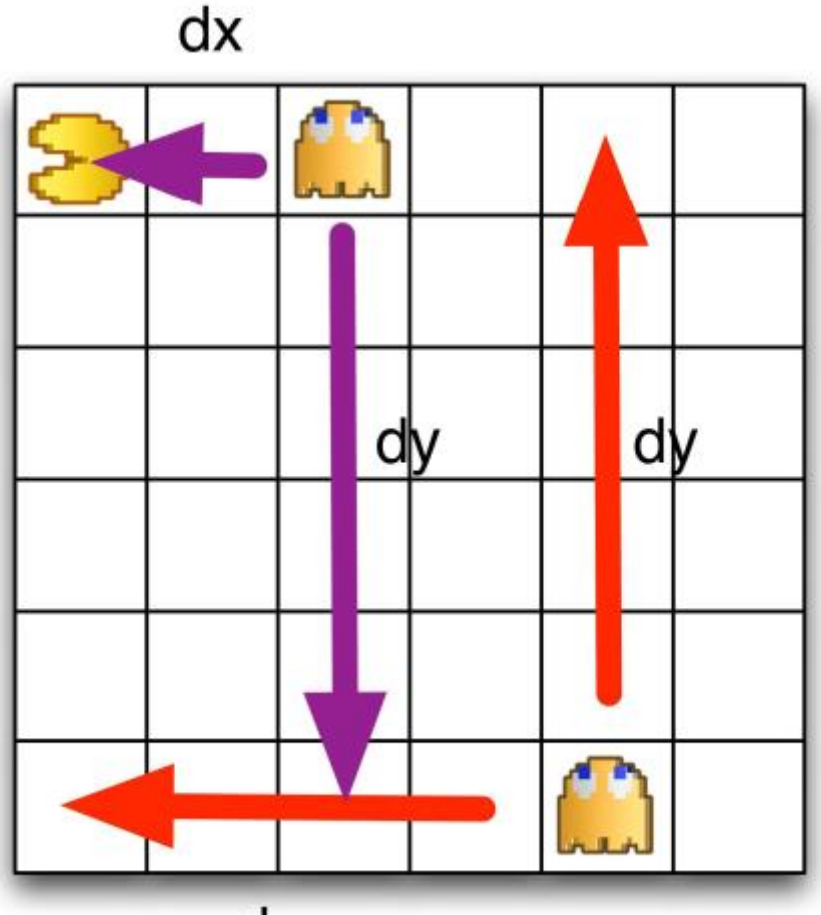
$$\vec{h} = max\left(\begin{pmatrix}\Delta x\\0\end{pmatrix},\begin{pmatrix}0\\\Delta y\end{pmatrix}\right)$$

Pokey



- Nickname: Clyde
- Pokey is the loner of the group, usually off doing his own thing when he’s not patrolling the southwest corner of the maze. His behavior is very random.
- Pokey has two basic Ais
  - one for when he’s far from Pac-Man (beyond 8 grid spaces):
    - Pokey behaves very much like Blinky trying to move to Pac-Man’s exact location
  - One for when he is near to Pac-Man:
    - Pokey goes to his home corner in the bottom left of the maze

Pokey



$$\vec{h} = max\left(\begin{pmatrix}\Delta x\\0\end{pmatrix},\begin{pmatrix}0\\\Delta y\end{pmatrix}\right)$$