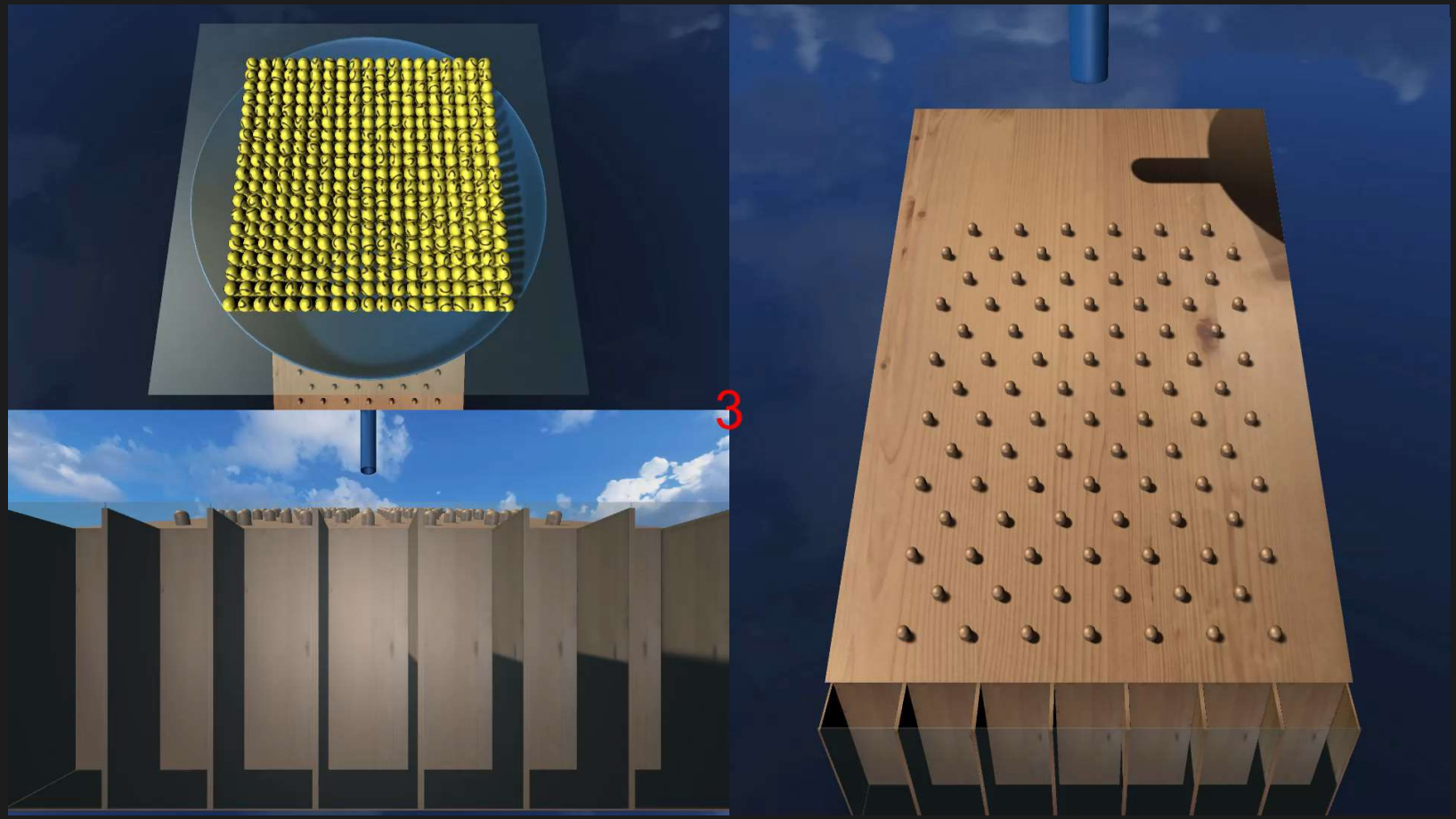
Einführung in Grundlagen der 3D Spielentwicklung

Ass.Prof. Dipl.-Ing. Dr.techn. Michael Urbanek, BA B.Sc. MSc

NEW DESIGN UNIVERSITY
PRIVATUNIVERSITÄT ST. PÖLTEN

Termine

Einheit	Datum	Zeit	Fokus der Einheit (work-in-progress-und-nicht-100-prozent-fix-:)
1.	06.05.2020	14:30 – 18:15	Formalitäten, Einführung Spiele, Game Design Prozess, Grundkenntnisse Game Engine & Mathematik, Einführung Unity3D, First Hands-on
2.	13.05.2020		Beginn ÜB I – Ball in the Maze (Camera, Input, Collision, Win Condition, Lose Condition)
3.	20.05.2020		Beginn ÜB II – Wahrscheinlichkeiten (Instantiate, Timer, Multiple Cams, Shader, Assets)
4.	27.05.2020		Game Design Document Beginn ÜB III – Pinball? (UnityUI, Animation, Sound, …)
5.	03.06.2020		Vorstellen Game Design Ideen + Diskussion Beginn ÜB IV – Shooter (Scenes, Menu, Raycast, Destroy)
6.	10.06.2020		Präsentation WIP Abschlussprojekt


3

Bleibt zuhause, spielt Spiele und helft so der Verbreitung entgegenzuwirken.

ECTS-Breakdown

ECTS	Stunden	Beschreibung
2	~ 50h	TOTAL
	22,5h	„Präsenz“einheiten
	0,5h	Besprechung Abschlussprojekt
	27h	Übungsblätter, Abschlussprojekt

Beurteilung

Übungsblätter I – IV
    jeweils 15% der Gesamtnote

Abschlussprojekt
    40% der Gesamtnote
    Gruppenarbeit

ggf. Anpassungen (zu Ihren Gunsten :-)
wegen COVID-19 und Distance Learning

A+	Sehr gut	100% - 98%
A		97% - 93%
A-		92% - 89%
B+	Gut	88% - 85%
B		84% - 80%
B-		79% - 76%
C+	Befriedigend	75% - 72%
C		71% - 67%
C-		66% - 63%
D+	Genügend	62% - 59%
D		58% - 54%
D-		53% - 50%
F	Nicht genügend	49 – 0%

Video-Tutorials bei Bedarf

Melden Sie Themen ein, zu denen Sie gerne mehr wissen würden

oder

Themen, die ich zu kurz oder schlecht erklärt habe :-)

➔ Hat keinerlei Auswirkung auf Ihre Note :-)

APE Prinzip für diese Lehrveranstaltung

Annähernd an der Lösung ist gut genug („Passt scho!“)

Es reicht, wenn Ihre Lösung in etwa dem Geforderten entspricht. Approximationen sind in der Spielentwicklung nichts Ungewöhnliches!

Perfektionismus ist mehr als gut genug

Nichts muss perfekt sein, nicht perfekt funktionieren, aber funktionieren. In der professionellen Spielentwicklung hat Perfektionismus auch selten Platz: es gibt eigene Rollen in der Spielentwicklung, die dann mal „stopp“ sagen.

Einfachheit ist die Regel

Setzen Sie Ihre Ideen so einfach wie möglich um!

Michael Urbanek

Michael Urbanek

Assistenzprofessur für Informatik

» **Doctor technicae** (Dr.techn.), TU Wien, focus on: Audio Games, 2020
» **Diplom-Ingenieur** (Dipl.-Ing.), TU Wien, focus on: Audio Games, 2015.
» **Master of Science** (MSc), Technikum Wien, focus on: Playtesting, 2013.
» **Bachelor of Science** (B.Sc.), Hochschule Furtwangen, Germany, 2012.
» **Bachelor of Arts in Business** (BA), FH des BFI Wien, 2012.

TU WIEN

AudiCom

FH University of Applied Sciences TECHNIKUM WIEN

FH CAMPUS WIEN UNIVERSITY OF APPLIED SCIENCES

Audio Games?

Audiogames, as opposed to video games are computer games who's [sic!] main output is sound rather than graphics.

Using sound, games can have dimensions of atmosphere, and possibilities for gameplay that don't exist with visuals alone, as well as providing games far more accessible to people with **all levels of sight**.

- audiogames.net

Audio in Video Games...

complementary or accentuating to the game experience

usually doesn't affect game functionality

important for immersion

Audio in Audio Games...

replaces UI elements and navigation

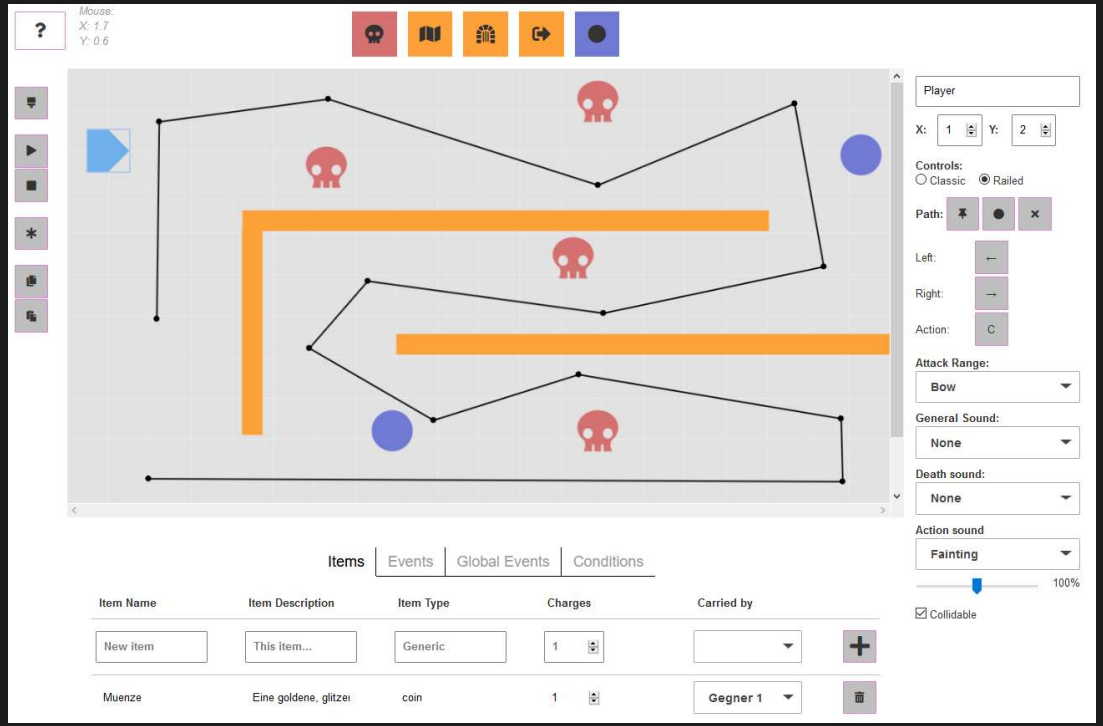
sonifies game objects; otherwise 'invisible'

feedback of player's actions

gameplot, mechanics, and meaning (immersion)

Manamon

AudiCom


Mouse:
X: 1.7
Y: 0.6
Player
X: 1
Y: 2
Controls:
Classic
Railed
Path:
Left:
Right:
Action:
C
Attack Range:
Bow
General Sound:
None
Death sound:
None
Action sound
Fainting
100%
Collidable
Items
Events
Global Events
Conditions
Item Name
Item Description
Item Type
Charges
Carried by
New item
This item...
Generic
1
Muenze
Eine goldene, glitzer
coin
1
Gegner 1

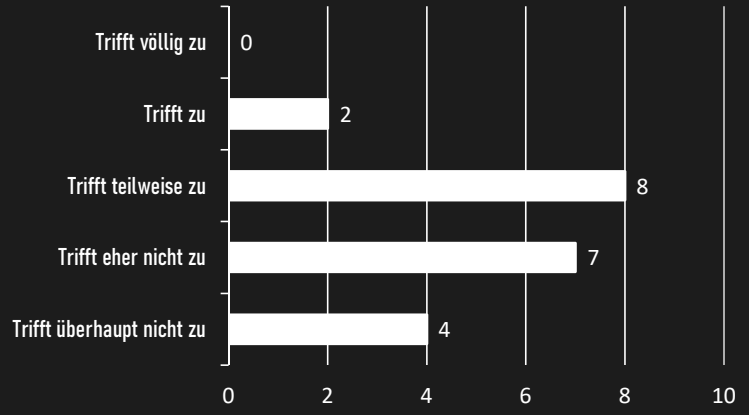
Übersicht

Heutiger Fahrplan

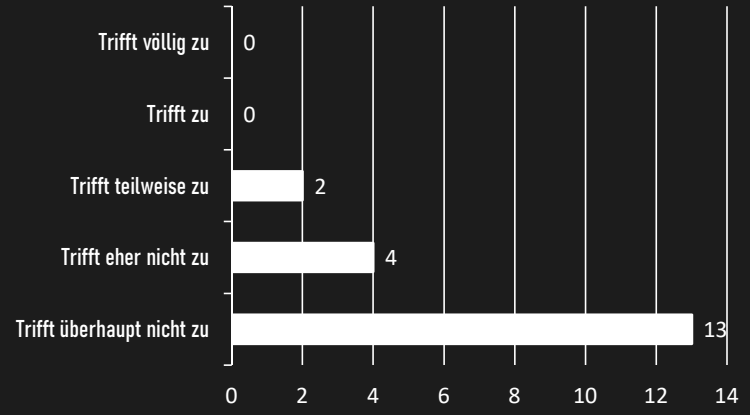
In etwa Zeit	In etwa Topics :-)
14:30 – 16:00	Quatschen, Kennenlernen, Foliensatz, Q&A
16:00 – 16:15	Pause, Stretching, Workout
16:15 – 18:15	Hands-on w/ Unity3D (with small tasks)

Fragebogen Ergebnisse (praktische Kenntnisse)

Ich habe praktische Kenntnisse in der Programmierung von Computern



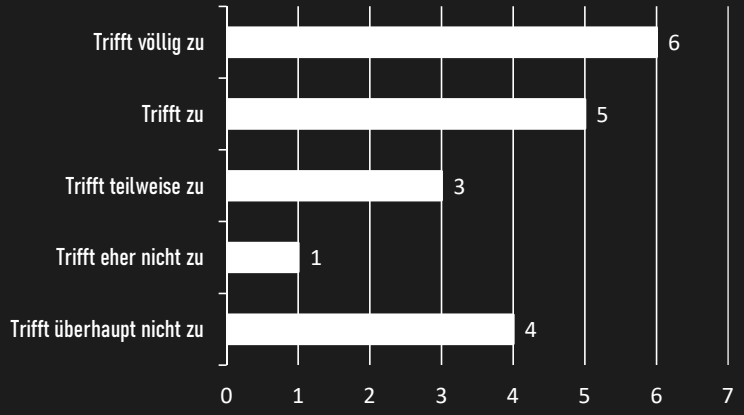
Ich habe praktische Kenntnisse in C oder C#



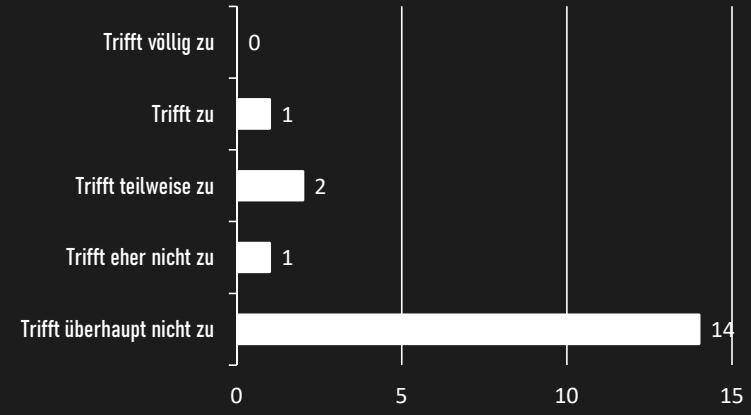
Processing, Python, JavaScript, HTML, CSS ...

Fragebogen Ergebnisse (praktische Kenntnisse)

Ich habe praktische Kenntnisse im Modellieren von 3D Objekten (z.B. Blender)



Ich habe praktische Kenntnisse im Arbeiten mit einer Game Engine



Blender, Cinema4D, Maya?

Fragebogen Ergebnisse

Rainbow 6: Siege
CoD:MW2
Minecraft
Child of Light
Metal Gear Solid series
Far Cry series
Super Animal Royale
Monument Valley
Overwatch
Witcher 3
Heroes of the Storm
Planet Coaster
Ori and the Blind Forest
Hearthstone
CS:GO
Path of Exile
Oxygen Not Included
Outlast
Zelda series
World of Warcraft
Fallout 4
Borderlands 3
Deep Rock Galactic
Metro series
League of Legends
City Skylines
Super Mario series
Satisfactory
XCOM series
Armello
Sims 4
Plague Inc.
Animal Crossing
Red Dead Redemption 2
Fortnite
Subnautica
Apex Legends
Factorio

Fragebogen Ergebnisse

Rainbow 6: Siege
CoD:MW2
Minecraft
Child of Light
Metal Gear Solid series
Far Cry series
Super Animal Royale
Monument Valley
Overwatch
Witcher 3
Heroes of the Storm
Planet Coaster
Ori and the Blind Forest
Hearthstone
CS:GO
Path of Exile
Oxygen Not Included
Outlast
World of Warcraft
Fallout 4
Borderlands 3
Deep Rock Galactic
Zelda series
League of Legends
City Skylines
Super Mario series
Metro series
Sims 4
Plague Inc.
Satisfactory
XCOM series
Armello
Fortnite
Animal Crossing
Red Dead Redemption 2
Apex Legends
Subnautica
Factorio

Fragebogen Ergebnisse

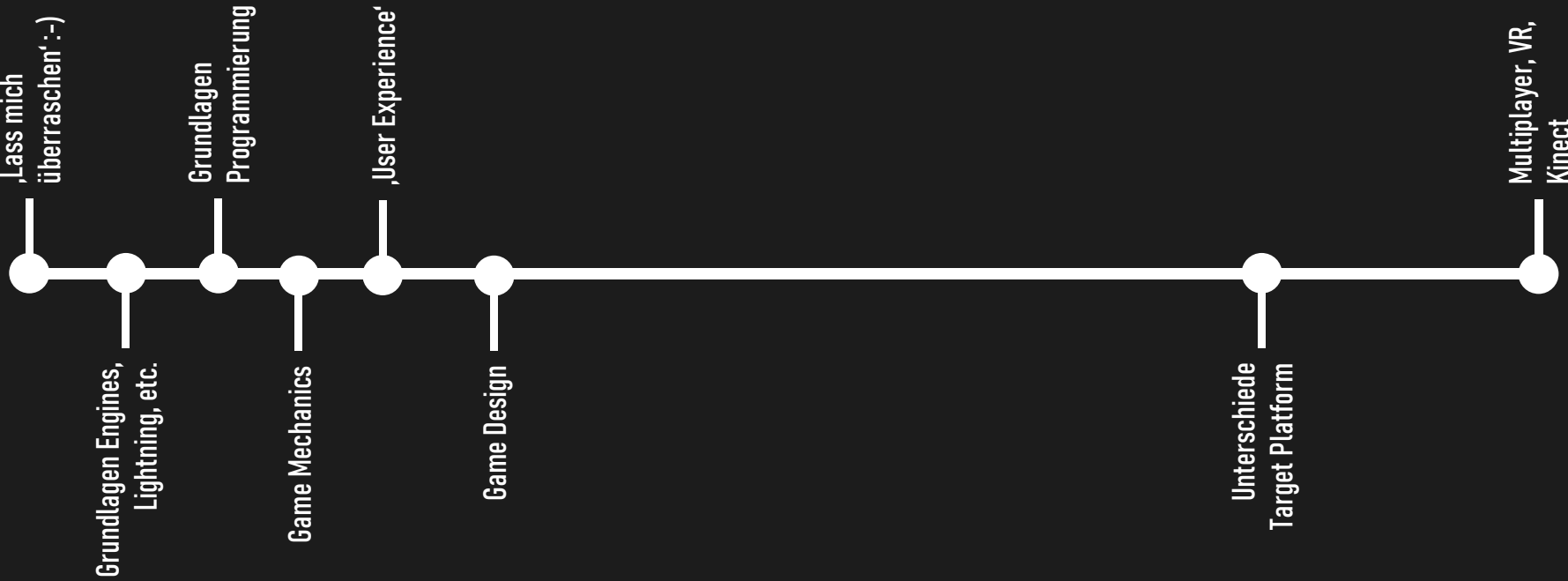
Rainbow 6: Siege
CoD:MW2
Minecraft
Child of Light
Metal Gear Solid series
Far Cry series
Super Animal Royale
Monument Valley
Overwatch
Witcher 3
Heroes of the Storm
Planet Coaster
Ori and the Blind Forest
Hearthstone
CS:GO
Path of Exile
Oxygen Not Included
Outlast
Zelda series
World of Warcraft
Fallout 4
Borderlands 3
Deep Rock Galactic
League of Legends
Sims 4
City Skylines
Super Mario series
Metro series
Armello
Plague Inc.
Satisfactory
XCOM series
Red Dead Redemption 2
Fortnite
Animal Crossing
Subnautica
Apex Legends
Factorio

Fragebogen Ergebnisse (Genres)

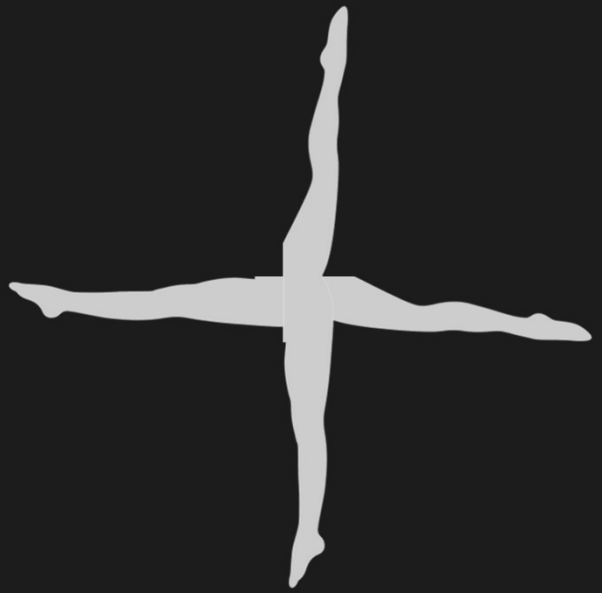
Indie Games, Platformer, story-driven, RPG, Puzzle, Puzzle Platformer, Metroidvania, Open World, Survival, Aufbausimulation, Action Adventure, Horror, Party Spiele, Quizspiele, Rollenspiele, Jump & Run, Virtual Reality, Survival, Strategie, Life Simulator, Shooter, Action, Point & Click, Action-Adventure, Turn-Based-Strategy, alles außer Strategie und Shooter, Simracing, Aufbau und Crafting, Action-Rollenspiele, Multiplayer, keine wirkliche Vorliebe, ...

... und ein Steam-Profil :-)

Fragebogen Ergebnisse (in LV vermittelt)



Game Design



Einführung Programmieren

Spielentwicklung (Unity3D)

2 ECTS

Fragebogen Ergebnisse (Liebe zu Mathematik :-)

AUSTRIAN GAME DEVELOPMENT LANDSCAPE 2019

AUSTRIAN GAME DEVELOPMENT LANDSCAPE 2019

WKO Unternehmensberatung · Buchhaltung · IT

DIRECT REVENUE IN 2016/2017

24.1 MILLION €

The revenue generated by domestic game developers adds up to 24.1 million € (direct effects).

OVERALL REVENUE IN 2016/2017

51.1 MILLION €

In addition, 13.2 million € are generated in companies (indirect effects) as well as 13.7 million € which are created via consumption (induced effects).

87 COMPANIES

Micro-enterprises and small enterprises shape the

470 DEVELOPERS

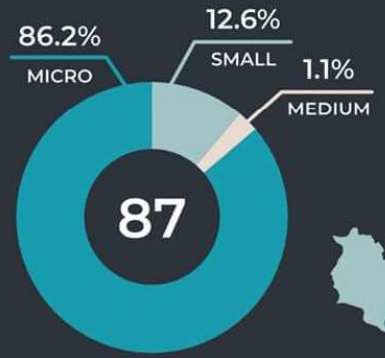
Every 7th developer expects an increase in revenue,

The revenue generated by domestic game developers adds up to 24.1 million € (direct effects).

In addition, 13.2 million € are generated in companies (indirect effects) as well as 13.7 million € which are created via consumption (induced effects).

87 COMPANIES

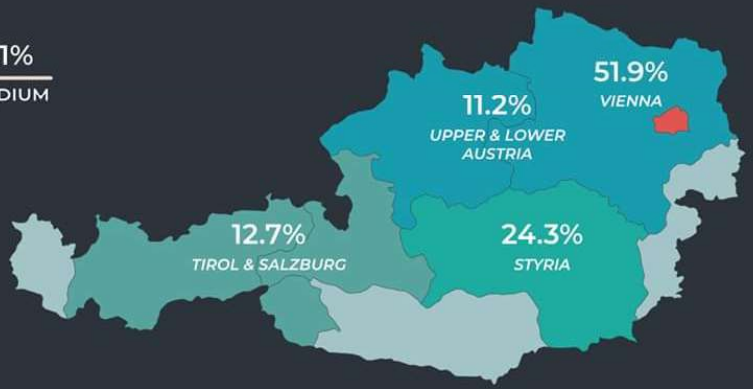
Micro-enterprises and small enterprises shape the corporate landscape of game development in Austria.



Micro = <10 employees
Small = 10-49 employees
Medium = 50-249 employees

470 DEVELOPERS

Every 7th developer expects an increase in revenue, whereas 57.1% expect a sales increase of more than 10%.



AGE DISTRIBUTION

Domestic game developers are younger when they settle in work life.

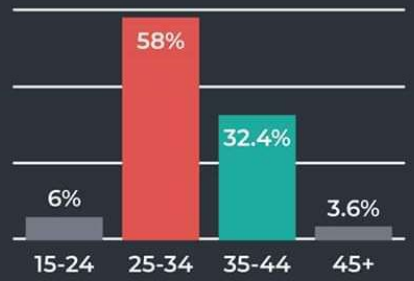
HIGHEST EDUCATION

More than 8 out of 10 respondents hold a university or applied university degree.

Micro = <10 employees
Small = 10-49 employees
Medium = 50-249 employees

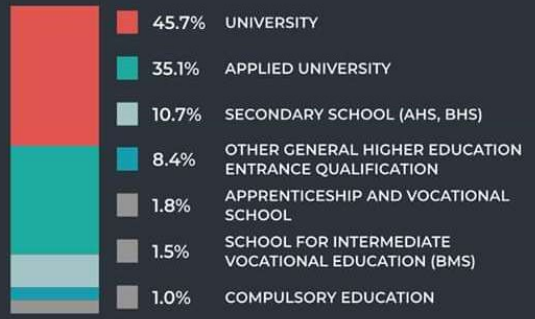
AGE DISTRIBUTION

Domestic game developers are younger when they settle in work life.

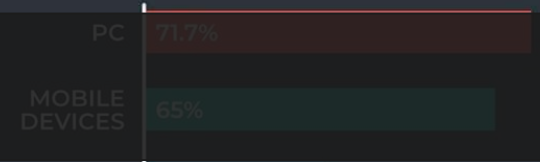


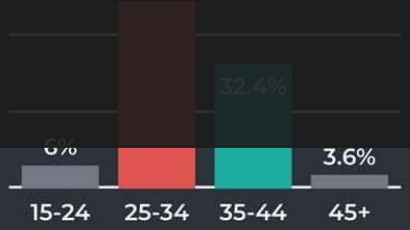
HIGHEST EDUCATION

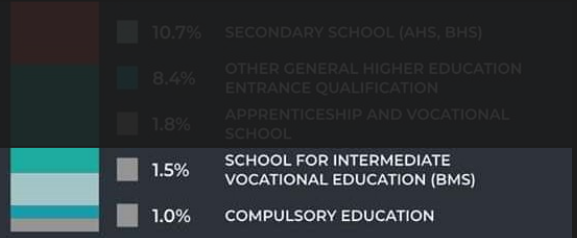
More than 8 out of 10 respondents hold a university or applied university degree.



DEVELOPMENT PLATFORM





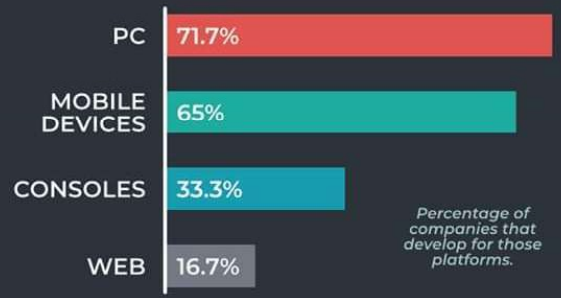


197

GAMES

MADE IN THE LAST 3 YEARS

DEVELOPMENT PLATFORM



Percentage of companies that develop for those platforms.

COMPANY DEVELOPMENT STATUS

Austrian game development companies are in an early stage of their life-cycle.

Most Austrian game developers choose the limited liability company (GmbH) as their preferred legal form.

GAMES

MADE IN THE LAST 3 YEARS

DEVICES 66.7%

CONSOLES 33.3%

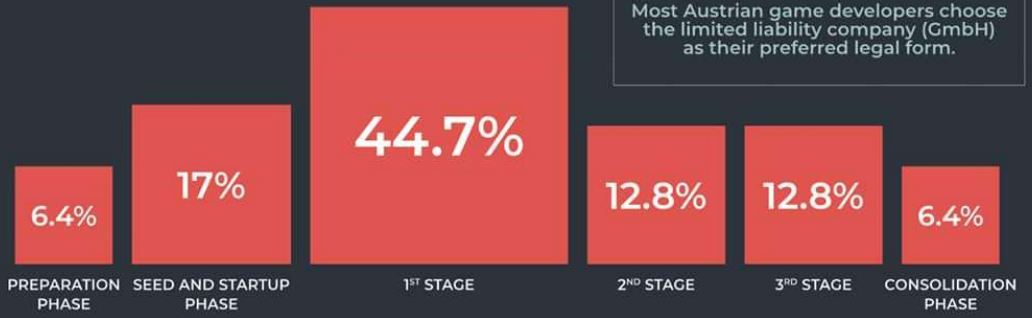
WEB 16.7%

Percentage of companies that develop for those platforms.

COMPANY DEVELOPMENT STATUS

Austrian game development companies are in an early stage of their life-cycle.

Most Austrian game developers choose the limited liability company (GmbH) as their preferred legal form.



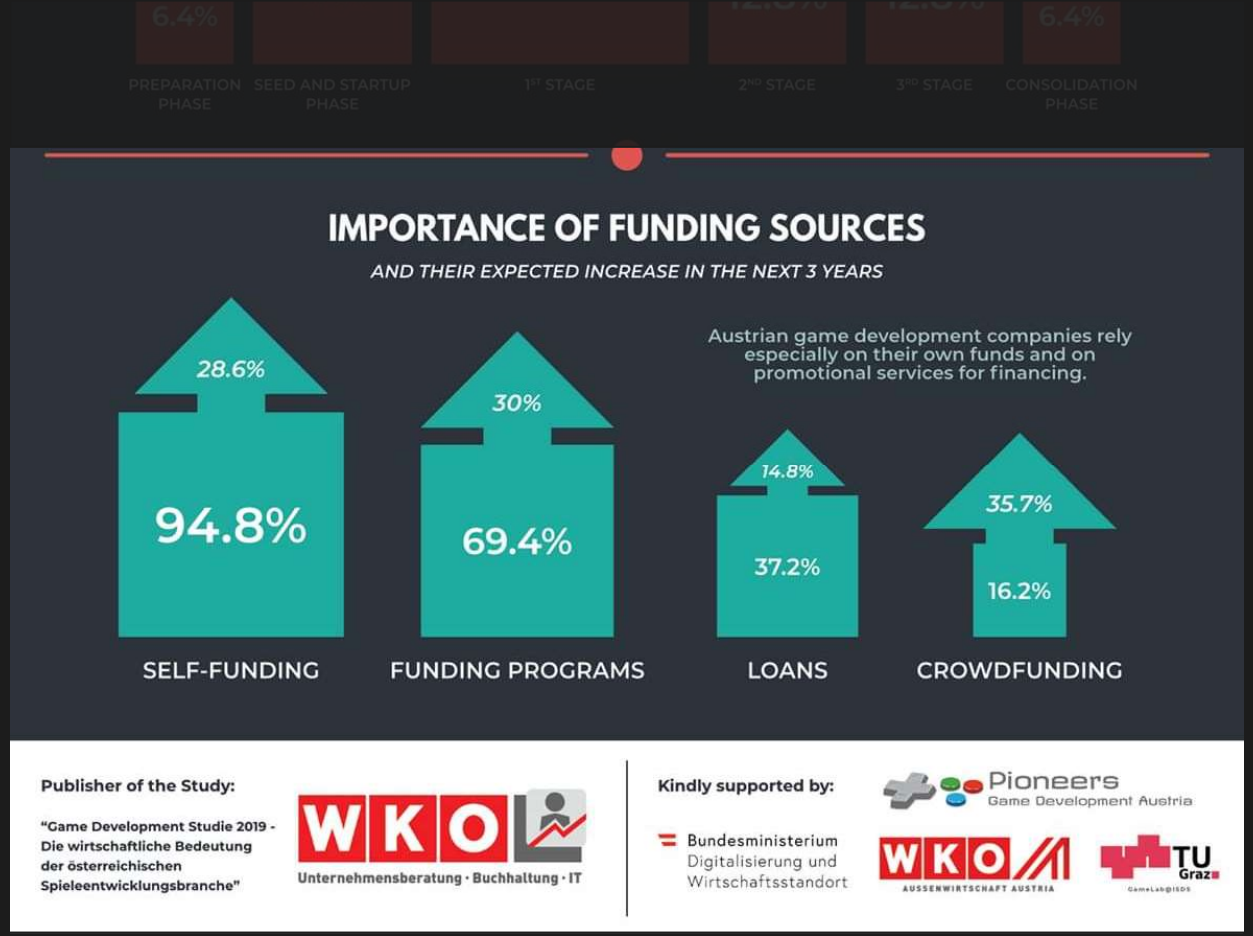
IMPORTANCE OF FUNDING SOURCES

AND THEIR EXPECTED INCREASE IN THE NEXT 3 YEARS

28.6%

30%

Austrian game development companies rely especially on their own funds and on promotional services for financing.


6.4%
PREPARATION PHASE
SEED AND STARTUP PHASE
1ST STAGE
2ND STAGE
3RD STAGE
CONSOLIDATION PHASE
6.4%
IMPORTANCE OF FUNDING SOURCES
AND THEIR EXPECTED INCREASE IN THE NEXT 3 YEARS
Austrian game development companies rely especially on their own funds and on promotional services for financing.
28.6%
94.8%
SELF-FUNDING
30%
69.4%
FUNDING PROGRAMS
14.8%
37.2%
LOANS
35.7%
16.2%
CROWDFUNDING
Publisher of the Study:
"Game Development Studie 2019 - Die wirtschaftliche Bedeutung der österreichischen Spieleentwicklungsbranche"
WKO
Unternehmensberatung · Buchhaltung · IT
Kindly supported by:
Pioneers
Game Development Austria
Bundesministerium
Digitalisierung und
Wirtschaftsstandort
WKO
AUSSENWIRTSCHAFT AUSTRIA
TU Graz

Themen

Was ist Spielen?

Game Design Prozess

Rollen im Game Design

Grundlagen der Programmierung

Grundlagen Mathematik in der Spielentwicklung

Grundlagen Game Engine

Was ist Spielen?

PLAY is older than culture, for culture, however inadequately defined, always presupposes human society, and animals have not waited for man to teach them their playing. We can safely assert, even, that human civilization has added no essential feature to the general idea of play. Animals play just like men. We have only to watch young dogs to see that all the essentials of human play are present in their merry gambols. They invite one another to play by a certain ceremoniousness of attitude and gesture. They keep to the rule that you shall not bite, or not bite hard, your brother's ear. They pretend to get terribly angry. And-what is most important-in all these doings they plainly experience tremendous fun and enjoyment. Such rompings of young dogs are only one of the simpler forms of animal play. There are other, much more highly developed forms: regular contests and beautiful performances before an admiring public. Here we have at once a very important point: even in its simplest forms on the animal level, play is more than a mere physiological phenomenon or a psychological reflex. It goes beyond the confines of purely physical or purely biological activity. It is a significant function-that is to say, there is some sense to it. In play there is something "at play" which transcends the immediate needs of life and imparts meaning to the action. All play means something. If we call the active principle that makes up the essence of play, "instinct", we explain nothing; if we call it "mind" or "will" we say too much. However we may regard it, the very fact that play has a meaning implies a nonmaterialistic quality in the nature of the thing itself.

Johann Huizinga – Homo Ludens

PLAY is older than culture, for culture, however inadequately defined, always presupposes human society, and animals have not waited for man to teach them their playing. We can safely assert, even, that human civilization has added no essential feature to the general idea of play. Animals play just like men. We have only to watch young dogs to see that all the essentials of human play are present in their merry gambols. They invite one another to play by a certain ceremoniousness of attitude and gesture. They keep to the rule that you shall not bite, or not bite hard, your brother's ear. They pretend to get terribly angry. And-what is most important-in all these doings they plainly experience tremendous fun and enjoyment. Such rompings of young dogs are only one of the simpler forms of animal play. There are other, much more highly developed forms: regular contests and beautiful performances before an admiring public. Here we have at once a very important point: even in its simplest forms on the animal level, play is more than a mere physiological phenomenon or a psychological reflex. It goes beyond the confines of purely physical or purely biological activity. It is a significant function-that is to say, there is some sense to it. In play there is something "at play" which transcends the immediate needs of life and imparts meaning to the action. All play means something. If we call the active principle that makes up the essence of play, "instinct", we explain nothing; if we call it "mind" or "will" we say too much. However we may regard it, the very fact that play has a meaning implies a nonmaterialistic quality in the nature of the thing itself.

Johann Huizinga - Homo Ludens

meaning ⇔ play

the goal of successful game design is the creation of meaningful play

Katie Salen, Eric Zimmerman – Rules of Play

meaningful play emerges from the interaction between players and the system of the game, as well as from the context in which the game is played

Katie Salen, Eric Zimmerman – Rules of Play

when a player makes a choice within a game, the action that results from the choice has an outcome

Katie Salen, Eric Zimmerman – Rules of Play

playing a game means making choices
and taking actions

Katie Salen, Eric Zimmerman – Rules of Play

meaningful play in a game emerges from the relationship between player action and system outcome; it is the process by which a player takes action within the designed system of a game and the system responds to the action. The meaning of an action in a game resides in the relationship between action and outcome

Katie Salen, Eric Zimmerman – Rules of Play

meaningful play occurs when the relationships between actions and outcomes in a game are both discernable and integrated into the larger context of the game

Katie Salen, Eric Zimmerman – Rules of Play

discernable

means that a player can perceive the immediate outcome of an action

Katie Salen, Eric Zimmerman – Rules of Play

integrated

means that the outcome of an action is woven into the game system as a whole

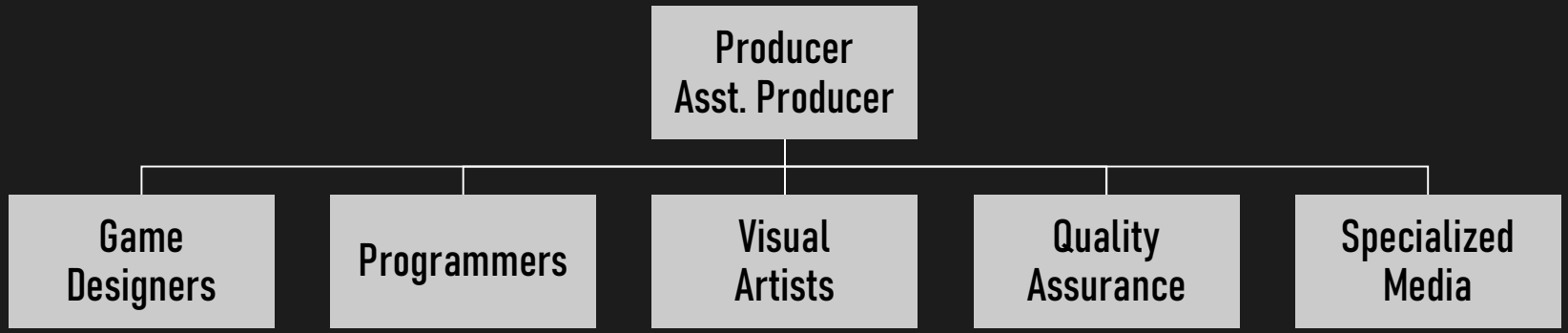
Katie Salen, Eric Zimmerman – Rules of Play

Game Design Prozess

Developer

Publisher

Developer


Producer
Asst. Producer
Game Designers
Programmers
Visual Artists
Quality Assurance
Specialized Media

Publisher

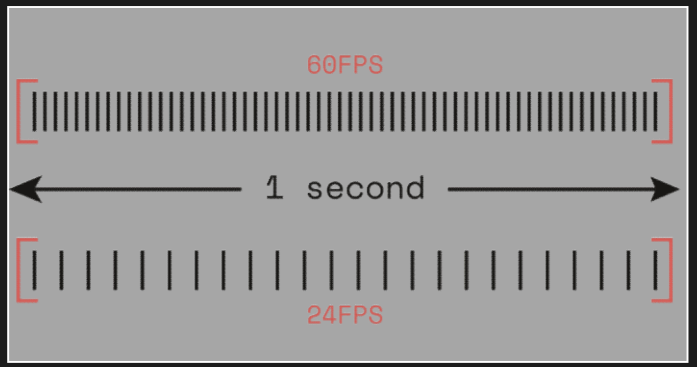

Producer
Asst. Producer
Executives
Marketing Team
Quality Assurance


Brainstorming
Physical Prototype
Presentation
Software Prototype(s)


...
Design Documentation
Production
Quality Assurance

Basics

Basics - FPS - Frames Per Second (oder Hz)

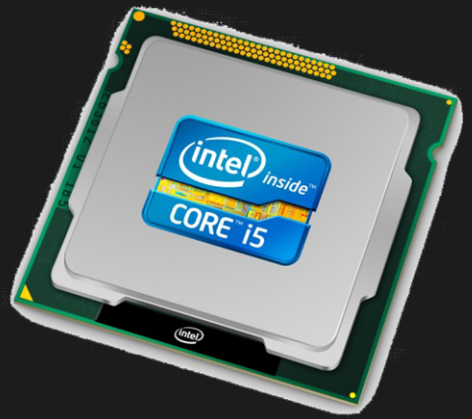


Anzahl an Bildern, die pro Sekunde angezeigt werden.

Menschliche Wahrnehmung: ab 24 Bildern flüssig

Filme – 24 Hz
3D Filme – 48 Hz
Spiele – 30 bis 240 Hz

Basics - CPU - Central Processing Unit



Das „Herz“ eines jeden Computers, zuständig für generelle Berechnungen

In 3D Spielentwicklung meistens NICHT für die grafische Darstellung zuständig, sondern für sequentielle Aufgaben, z.B. AI.

Je mehr Kerne, umso mehr Aufgaben können parallel berechnet werden (multi-tasking).

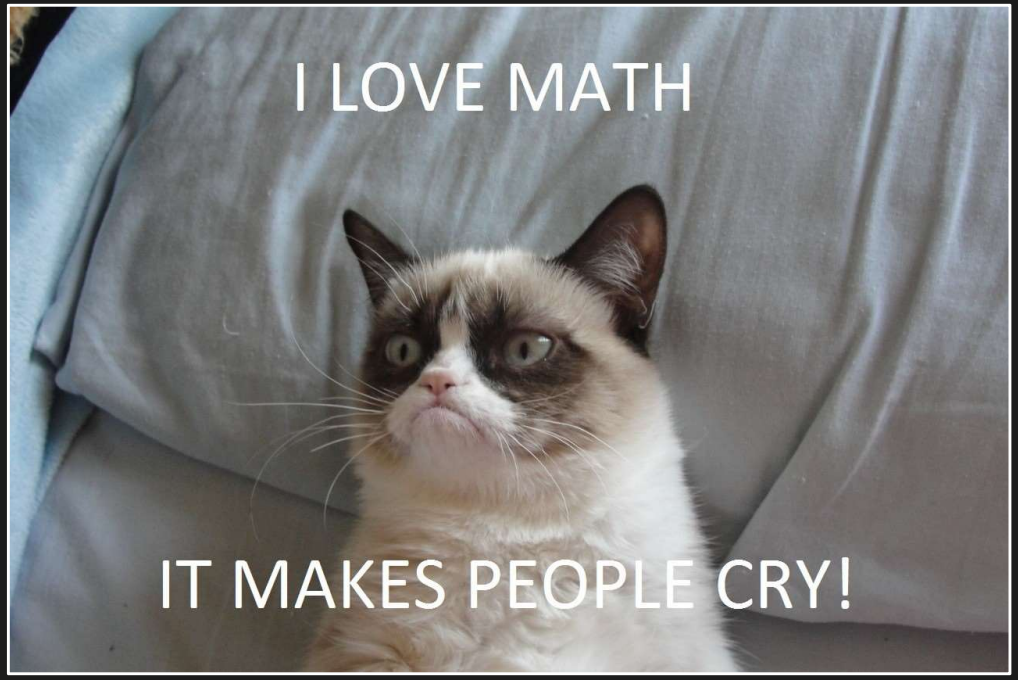
Basics - GPU - Graphics Processing Unit

Das „andere Herz“ eines Computers, zuständig für parallele Berechnungen

Dabei meistens für grafische, parallele Berechnungen verwendet

Mehrere tausend Kerne, die gleichzeitig arbeiten (Bsp.: RTX 2080 Ti, 4352 Kerne)

Basics – Mathematik


I LOVE MATH
IT MAKES PEOPLE CRY!

Koordinatensystem, Vektoren

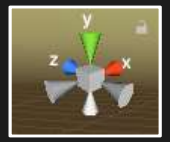
Mathematisches Objekt, das eine Parallelverschiebung in der Ebene oder im Raum beschreibt.

Ein Vektor kann durch einen Pfeil, der einen Urbildpunkt mit seinem Bildpunkt verbindet, dargestellt werden.

Vektoren können **addiert** und mit reellen Zahlen (Skalaren) **multipliziert** werden.

Motiviert von der Koordinatendarstellung der geometrischen Vektoren werden oft auch n-Tupel reeller Zahlen, also Elemente des $\mathbb{R}^n$, als Vektoren oder auch als Koordinatenvektoren bezeichnet.

3D Vektoren



… bestehen also aus 3 reellen Zahlen (x,y,z)

… stellen eine Koordinate im 3D-Raum dar

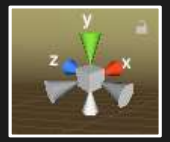
… wird auch benutzt, um bspw. Geschwindigkeiten darzustellen

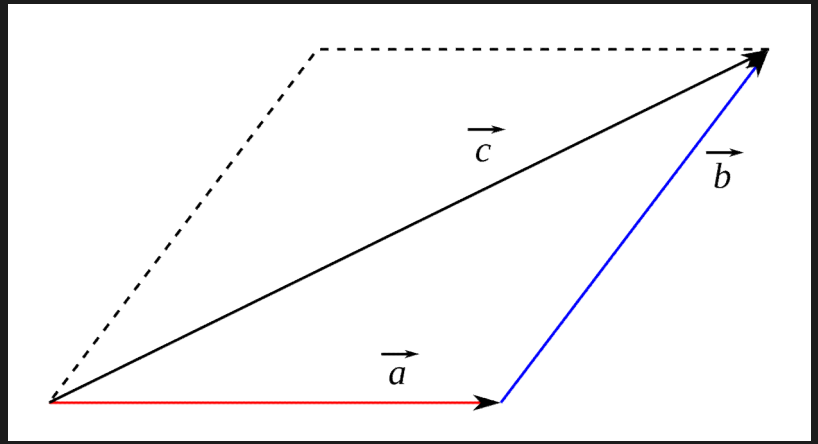
… ermöglicht Rechenoperationen wie Addition, Subtraktion, Multiplikation mit Skalar, Skalarprodukt, Kreuzprodukt, Betrag

Datentyp in Unity3D: **Vector3**
(https://docs.unity3d.com/ScriptReference/Vector3.html)

C#
```
Vector3 koord = new Vector3(0f, 1f, 0f);
```

3D Vektoren - Addition

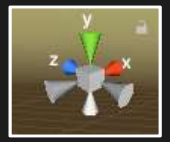


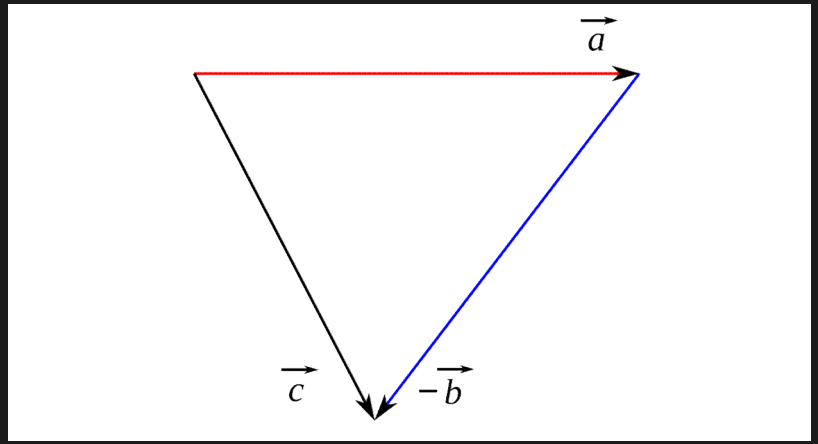


C#

```
Vector3 a = new Vector3(2f, 0f, 0f);
Vector3 b = new Vector3(1f, 2f, 0f);
Vector3 c = a + b;
```

3D Vektoren - Subtraktion

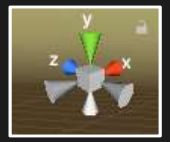


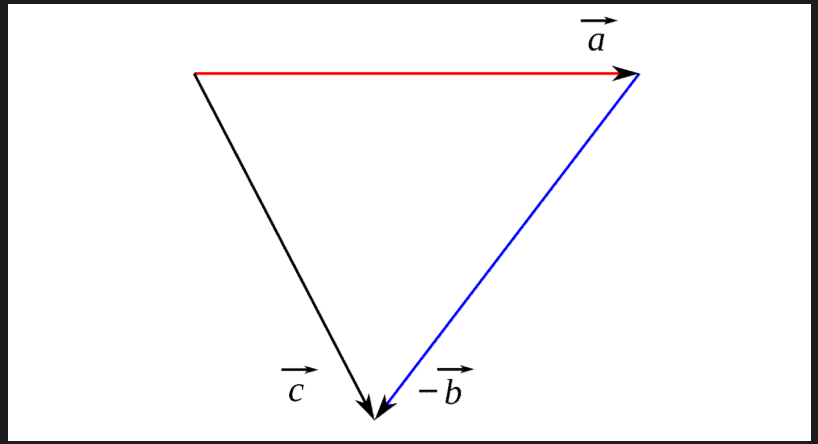


C#

```
Vector3 a = new Vector3(2f, 0f, 0f);
Vector3 b = new Vector3(1f, 2f, 0f);
Vector3 c = a - b;
```

3D Vektoren - Subtraktion

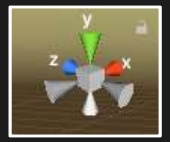


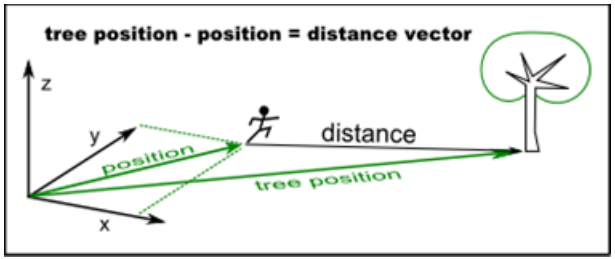


C#

```
Vector3 a = new Vector3(2f, 0f, 0f);
Vector3 b = new Vector3(-1f, -2f, 0f);
Vector3 c = a + b;
```

3D Vektoren - Subtraktion - Beispiel

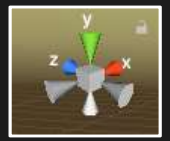


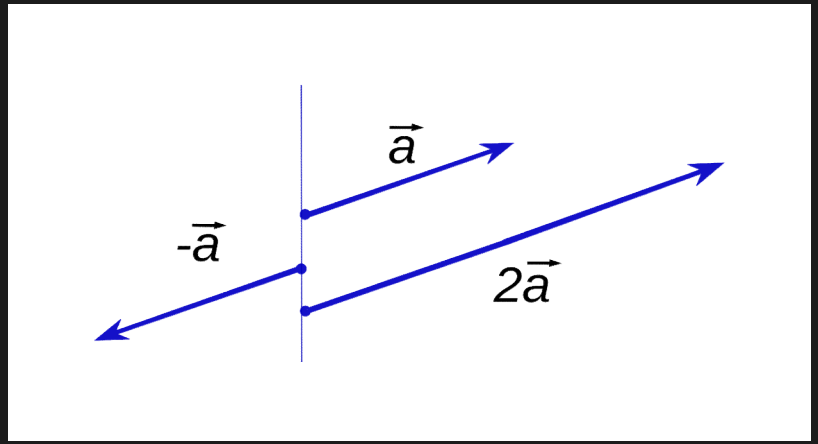


C#

```
Vector3 tree = new Vector3(10f, 2f, 0f);
Vector3 player = new Vector3(2f, 2f, 0f);
Vector3 distance = tree - player;
```

3D Vektoren - Multiplikation mit Skalar

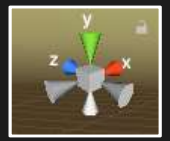


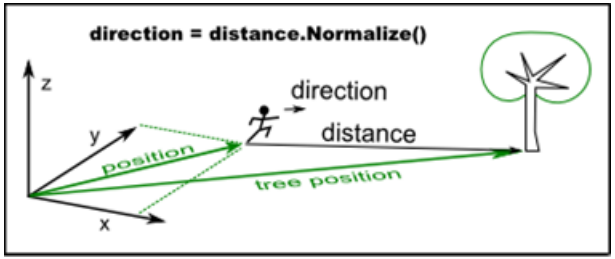


C#

```
Vector3 a = new Vector3(1f, 0f, 0f);
Vector3 c = -a;
c = 2 * a;
```

3D Vektoren - Betrag / Richtungsvektor / Länge 1

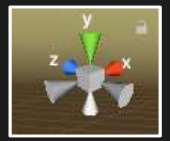


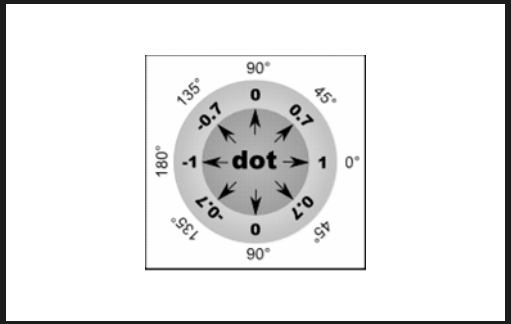


C#

```
Vector3 tree = new Vector3(10f, 2f, 0f);
Vector3 player = new Vector3(2f, 2f, 0f);
Vector3 distance = tree - player;
Vector3 direction = distance.normalized;
```

3D Vektoren - Skalarprodukt





C#

```
Vector3 forward = transform.TransformDirection(Vector3.forward);
Vector3 toOther = other.position - transform.position;

if (Vector3.Dot(forward, toOther) < 0)
print("The other transform is behind me!");
```

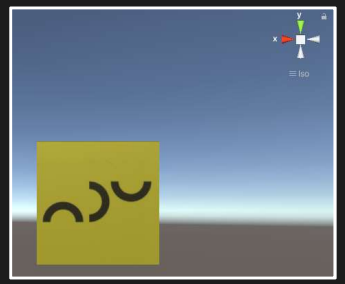
3D Vektoren - Affine Transformationen

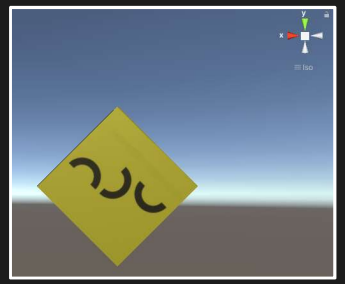
$$R_z(\phi)=\begin{bmatrix}\cos\phi & -\sin\phi & 0 & 0\\ \sin\phi & \cos\phi & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

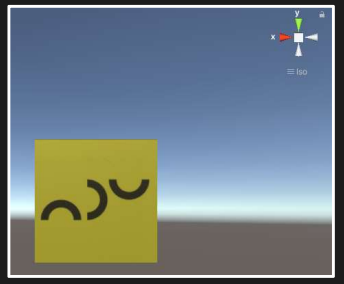
Rotation um z

$$S(k_x,k_y,k_z)=\begin{bmatrix}k_x & 0 & 0 & 0\\ 0 & k_y & 0 & 0\\ 0 & 0 & k_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

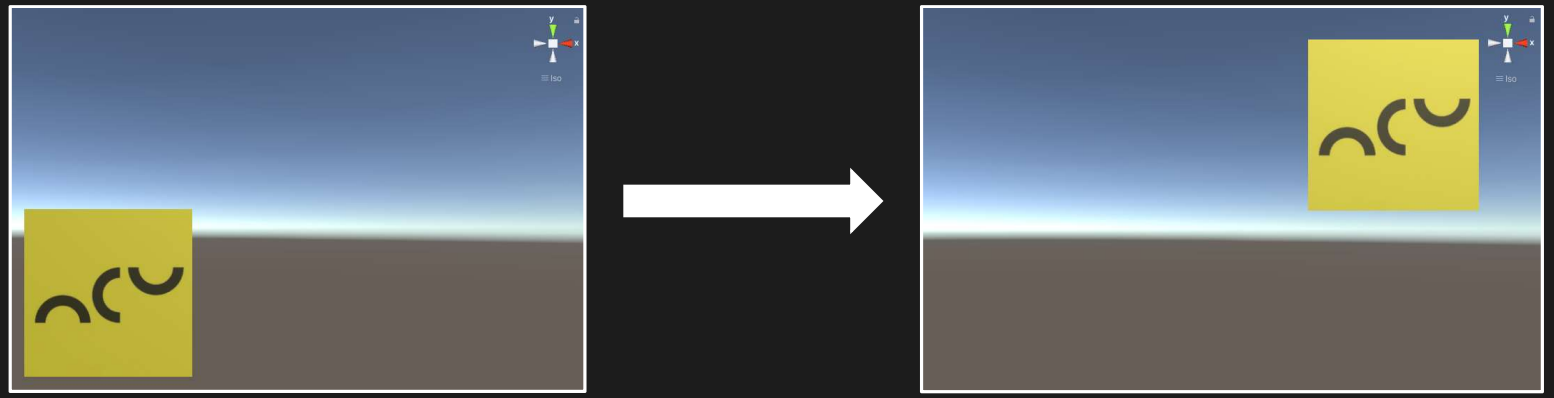
Skalieren







3D Vektoren - Affine Transformationen



$$T(k_x, k_y, k_z) = \begin{bmatrix} 1 & 0 & 0 & k_x \\ 0 & 1 & 0 & k_y \\ 0 & 0 & 1 & k_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Basics – Game Engine

Game Engine … wieso überhaupt?

Sie müssen sich um viele Dinge nicht kümmern (Shading, Physics, …)

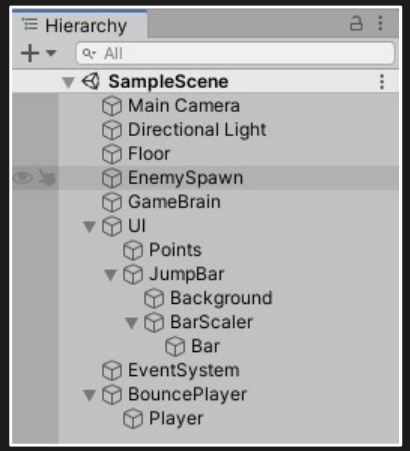
Zeigt Ihnen typische, notwendige Funktionalitäten an

Stellt eine getestete Umgebung zur Verfügung

Erlaubt das Exportieren auf verschiedene Zielplattformen

… dennoch müssen Sie den Rest, für Ihr Spiel, programmieren :-)

Szenegraph



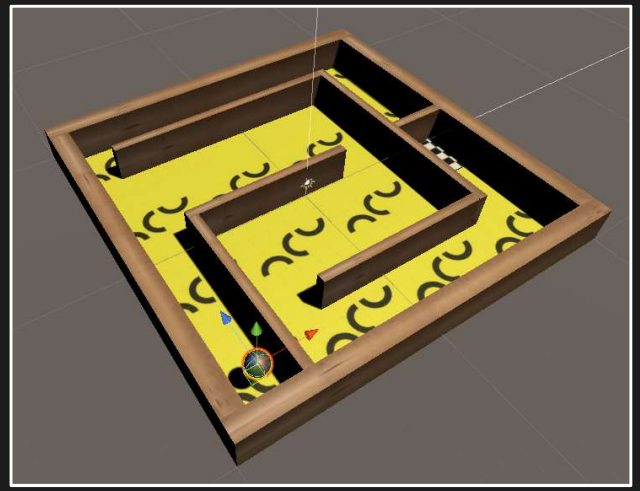
Ein Szenengraph ist eine Datenstruktur, die häufig bei der Entwicklung computergrafischer Anwendungen eingesetzt wird.

Aus graphentheoretischer Sicht ist ein Szenengraph ein zusammenhängender gerichteter Graph ohne gerichtete Kreise, dessen Wurzelknoten die Gesamtszene (das „Universum“) enthält. Dieser Wurzel untergeordnet sind Kindknoten, die die einzelnen Objekte der Szene, oder Eigenschaften wie Transformationen und Farben enthalten

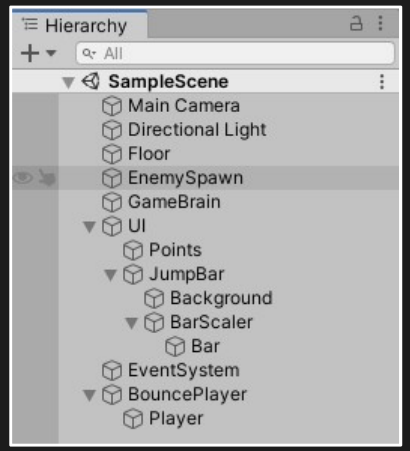
Szenegraph

Hierarchische Modellierung der Objekte in einer Szene.

Jeder Knoten des Szenengraphen hat eine Transformationsmatrix.



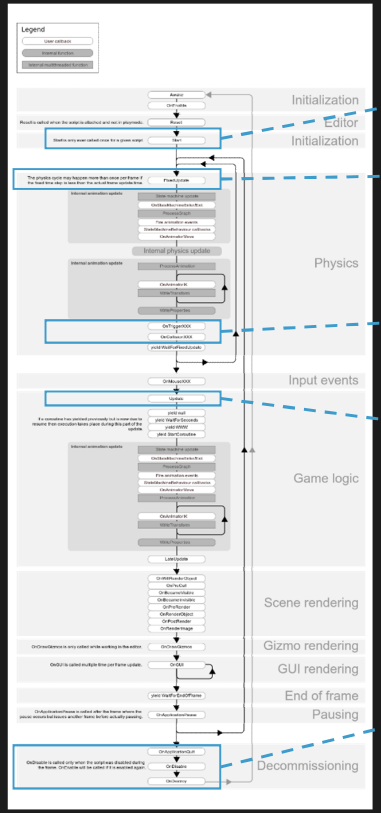
Szenegraph



Bei Manipulation dieser Matrix wird das zugehörige Objekt selbst, aber auch die Objekte aller untergeordneten Knoten transformiert. Man unterscheidet zwischen Objektkoordinaten (**lokale Koordinaten**) und Weltkoordinaten (**globale Koordinaten** eines Objektes bezüglich des Ursprungs des Universums – der Wurzel des Szenengraphen).

Durch diese hierarchische Sicht wird der Aufbau und das Manipulieren einer Szene deutlich vereinfacht. Man muss nicht jedes Einzelteil eines Objektes einzeln transformieren, sondern transformiert einfach die Gesamtheit aller Einzelteile.

Execution Order



Start is only ever called once for a given script.

Start

The physics cycle may happen more than once per frame if the fixed time step is less than the actual frame update time.

FixedUpdate

OnTriggerXXX

OnCollisionXXX

Update

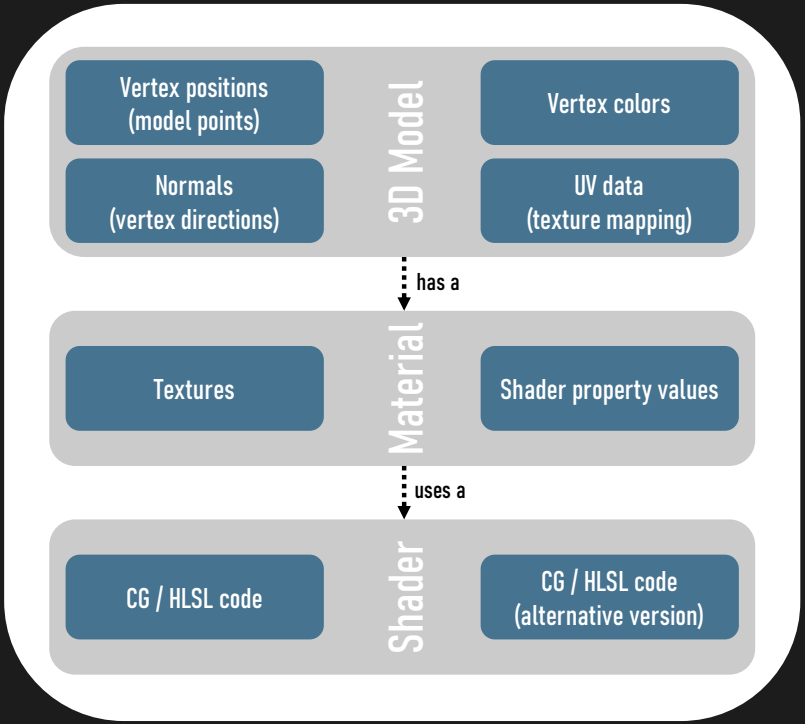
OnDisable is called only when the script was disabled during the frame. OnEnable will be called if it is enabled again.

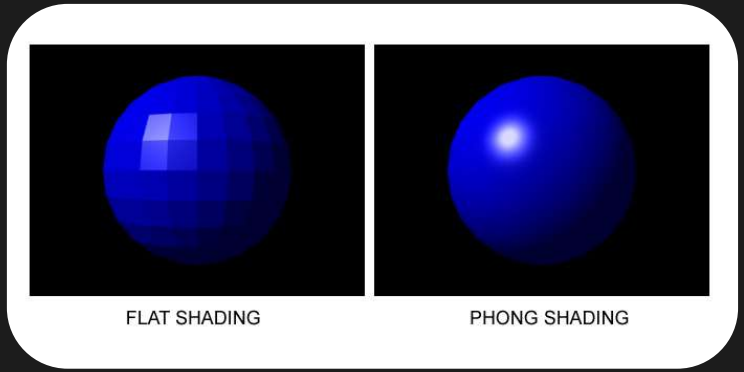
OnApplicationQuit

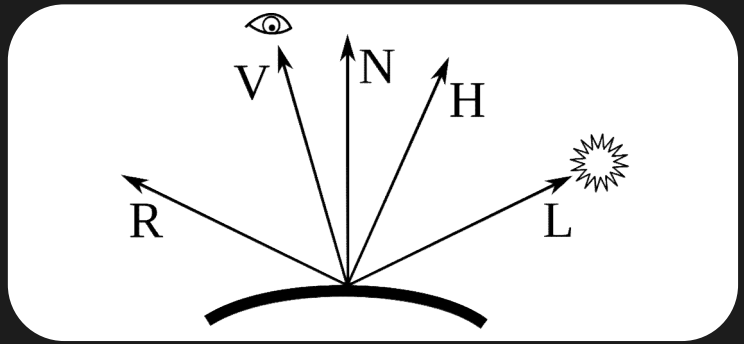
OnDisable

OnDestroy

Rendering in Unity

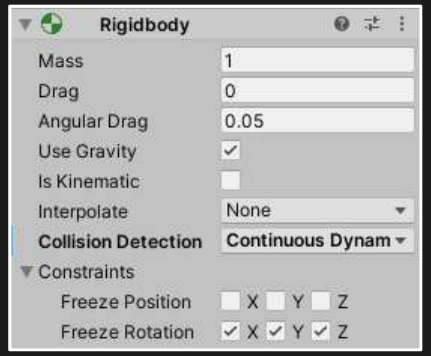


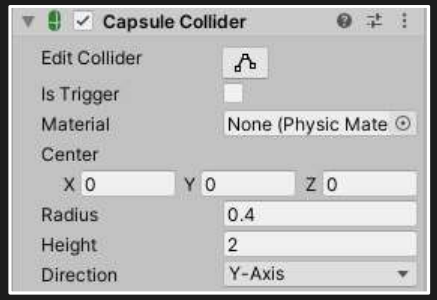




Blinn-Phong reflection model

Physics





Unity kümmert sich um das physikalisch-halbwegs-korrekte Verhalten von Objekten, die der Physikengine bekannt gemacht worden sind (Rigidbody!).

Basics – Programmieren

Programmieren - Allgemein

Programmieren brauchen wir für ...

- ... das Regeln der Abläufe,
- ... das setzen von Bedingungen,
- ... zum Ändern von Zuständen,
- ... zum iterativen Überprüfen der Bedingungen,
- ...

Auch wenn Sie relativ viel programmieren müssen, bedenken Sie, dass Unity3D im Hintergrund noch viel mehr übernimmt, über dass Sie sich keine Sorgen machen müssen!

Programmieren - Begriffe

Variable	. . . . .	‚Behälter', in dem Werte gespeichert werden
Datentyp	. . . . .	Art der Dinge, die im ‚Behälter' gespeichert werden
Integer, Float, String, Bool	. . . . .	Verschiedene Datentypen
Operator	. . . . .	+ - * / % && || ! == != <= >= < > << >> & | ^
Sichtbarkeit	. . . . .	Von wo aus ich auf Variablen etc. zugreifen kann
Ausdruck	. . . . .	Etwas, das wahr oder falsch sein kann
Wahr – Falsch	. . . . .	1 oder 0 :-)
Bedingung	. . . . .	Wenn erfüllt, dann wahr sonst falsch
Schleife	. . . . .	Ein Codeteil, das immer wieder ausgeführt wird
Funktion	. . . . .	Ein ausgelagertes Stück Code,
Rückgabeparameter	. . . . .	das etwas zurückliefern kann
Funktionsparameter	. . . . .	und etwas entgegennehmen kann.
Debuggen	. . . . .	Schrittweises Durchgehen der Codeausführung

Programmieren - Datentypen

C#

Bool	Kann ‚true' oder ‚false' sein	`bool _gameOver = false;`
Integer	speichert eine ganze Zahl	`int maxAmmo = 20;`
Float	speichert eine Gleitkommazahl	`float timeBeforeFading = 6.0f;`
String	speichert eine Zeichenkette	`string name = "Player";`
Vector3	speichert drei Gleitkommazahlen	`Vector3 tree = new Vector3(10f, 2f, 0f);`
GameObject	speichert eine Referenz auf ein Objekt	`GameObject player = GameObject.FindGameObjectWithTag("Player");`
Transform	speichert Position, Rotation und Scale	`Transform pTrans = player.transform;`

Programmieren - Operatoren

+	. . . . .	Addition und String / Zeichenverkettung
-	. . . . .	Subtraktion
*	. . . . .	Multiplikation
/	. . . . .	Division
%	. . . . .	Restklassenrechnung Modulo
==	. . . . .	gleich
!=	. . . . .	ungleich
!	. . . . .	negieren
< >	. . . . .	kleiner bzw. größer
<= >=	. . . . .	kleiner gleich bzw. größer gleich
&&	. . . . .	und
||	. . . . .	oder

Programmieren - Operatoren Beispiele

+	„Peter“ + „ ist lustig“	. . . . .	-
-	14 - 3	. . . . .	-
*	3 * 5 * 4	. . . . .	-
/	4 / 3	. . . . .	-
%	4 % 3	. . . . .	-
==	4 == 4	. . . . .	WAHR
!=	4 != 4	. . . . .	FALSCH
!	!(4 == 4)	. . . . .	FALSCH
< >	4 < 3	. . . . .	FALSCH
<= >=	3 >= 3	. . . . .	WAHR
&&	(3 + 3) && (4 < 1)	. . . . .	FALSCH
||	(3 + 3) || (4 < 1)	. . . . .	WAHR

Programmieren - Bedingungen / Ausdrücke

„Spieler“ == „Spieler“	. . . . .	WAHR
„Spieler“ == „spieler“	. . . . .	FALSCH
4 % 4 == 0	. . . . .	WAHR
4 % 3 != 0	. . . . .	WAHR
(5 < 4) && (2 >= 2)	. . . . .	FALSCH
((5 < 4) && (2 >= 2) || true)	. . . . .	WAHR
10 / 5 == 2	. . . . .	WAHR
5 + 2 * 5 == 25	. . . . .	WAHR
true != true	. . . . .	FALSCH
!(true != true)	. . . . .	WAHR
!true && (true != true)	. . . . .	FALSCH

Programmieren - Verzweigungen

Eine Verzweigung innerhalb eines Programmes oder Codesegments wird durch eine Bedingung entschieden. Ist die Bedingung wahr, wird ein Zweig (Codeblock) ausgeführt, andernfalls der andere.

```
if(2 > 3)
{
    Debug.Log(“2 ist größer als 3!”);
} else
{
    Debug.Log(
        “2 ist nicht größer als 3!”);
}
```

```
if(gameObject.name == „Player“)
{
    Debug.Log(“Spieler gefunden!”);
} else
{
    Debug.Log(“Unbekanntes Objekt!”);
}
```

Programmieren - Schleifen

Das iterative Ausführen von einem Codestück bis eine Abbruchbedingung erfüllt ist. Ist die Abbruchbedingung nicht erfüllt oder gibt es keine Abbruchbedingung, spricht man von einer Endlosschleife.

```
transform.position = new Vector3(0f, 0f, 0f);

for(int i = 0; i < 10; i++)
{
    transform.position = new Vector3(
        transform.position.x + i,
        transform.position.y,
        transform.position.z);
}
```

```
while (true)
{
    transform.position = new Vector3(
        transform.position.x + 1,
        transform.position.y,
        transform.position.z);
}
```