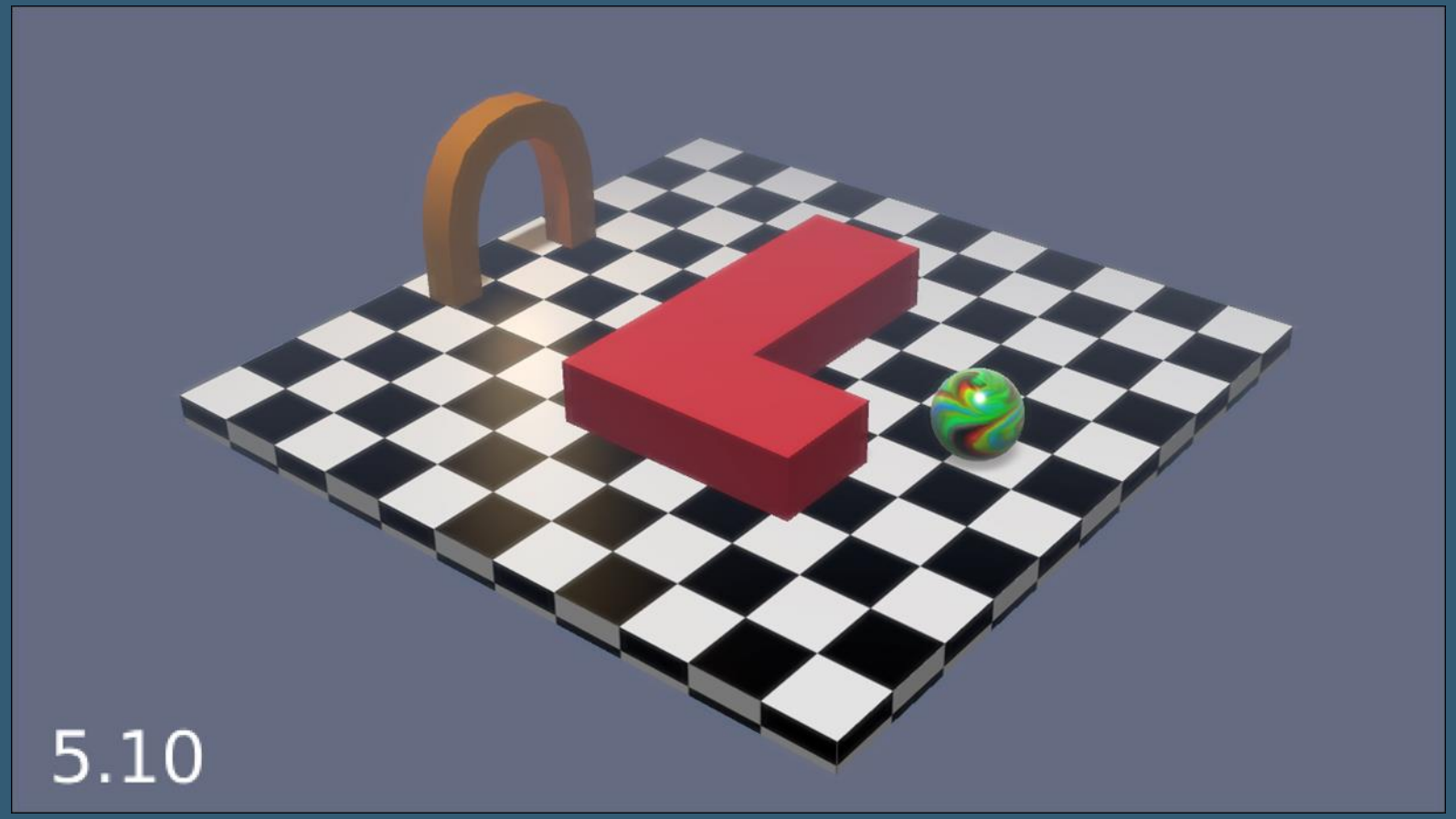
BLENDER GAME A

Tutorium von Raphael Menges

ZIEL



THEMEN

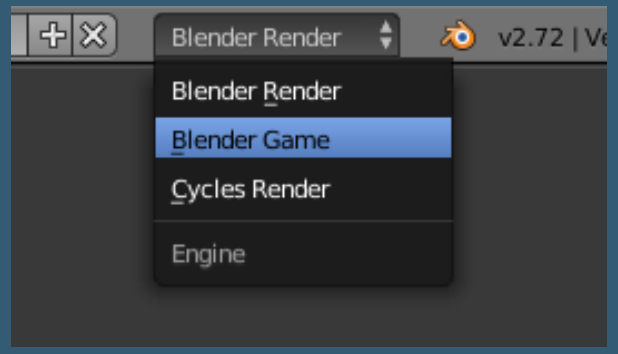
- Einführung
- Spielwelt
- Materialien
- Physik
- Gameplay
- Deployment

BLENDER GAME ENGINE (BGE)

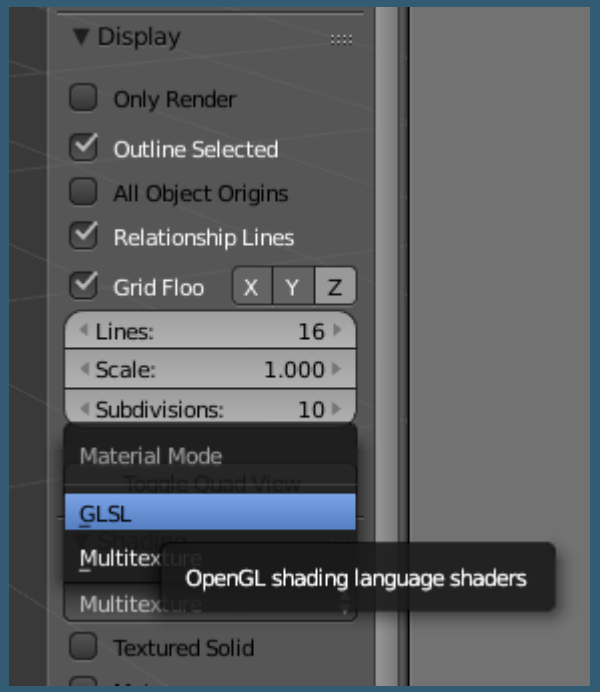
- Game Engine direkt in Blender
- In C++ geschrieben
- Python Schnittstelle für Spiellogik
- Fertige Logikblöcke zum Verbinden
- Einige Blender-Systeme funktionieren auch im Spiel (Armature, Physik teilweise)
- Kann auf Windows, Linux und Mac OS deployed werden
- Sehr gut für Prototypen geeignet, da innerhalb von Blender
- Kommerzielle Projekte wegen GPL schwierig

http://www.yofrankie.org/

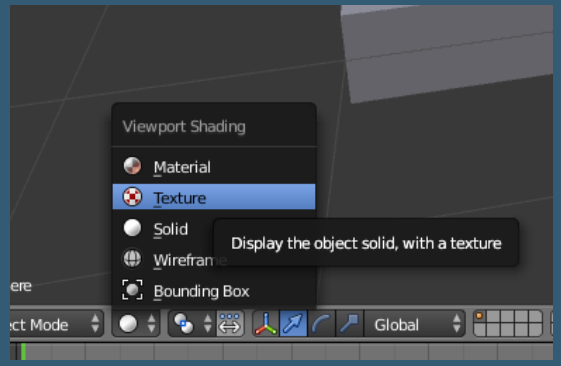
SETUP



In Leiste ganz oben



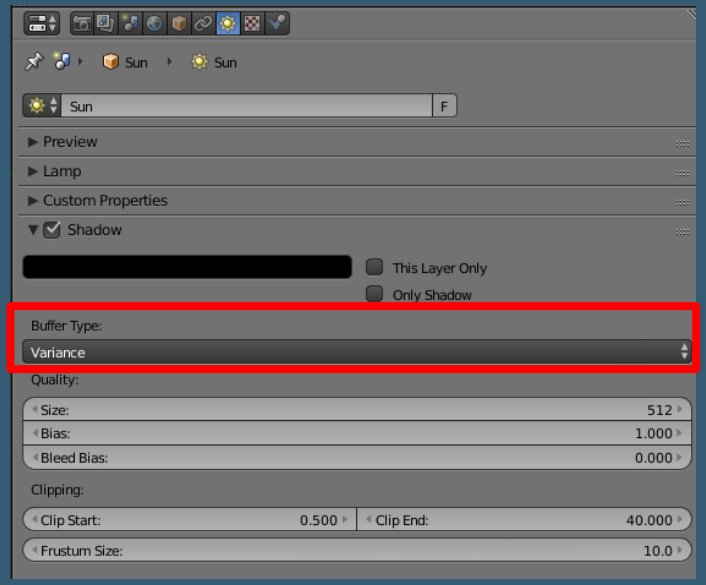
In Properties der 3D-View

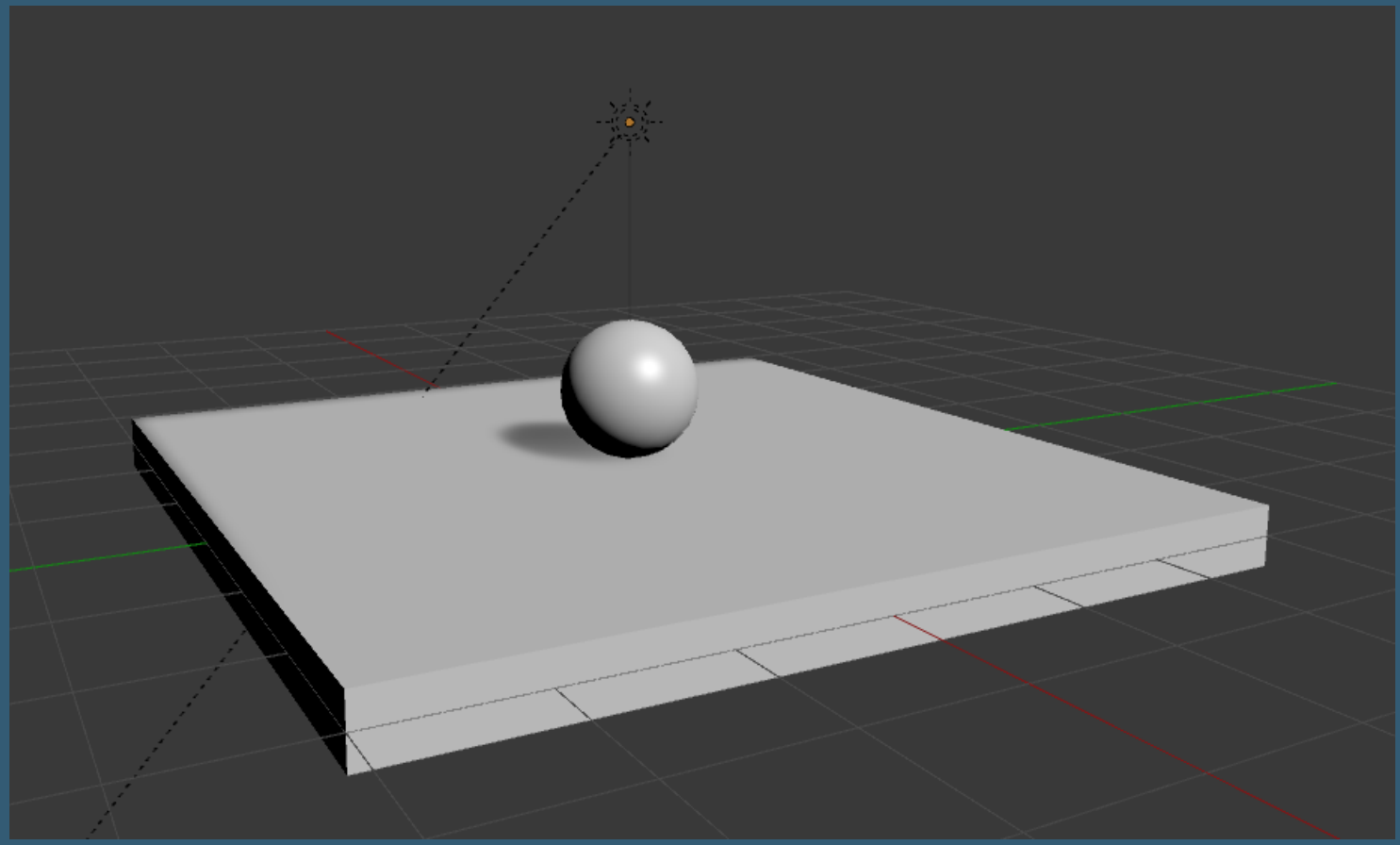


In 3D-View unten

ASSETS

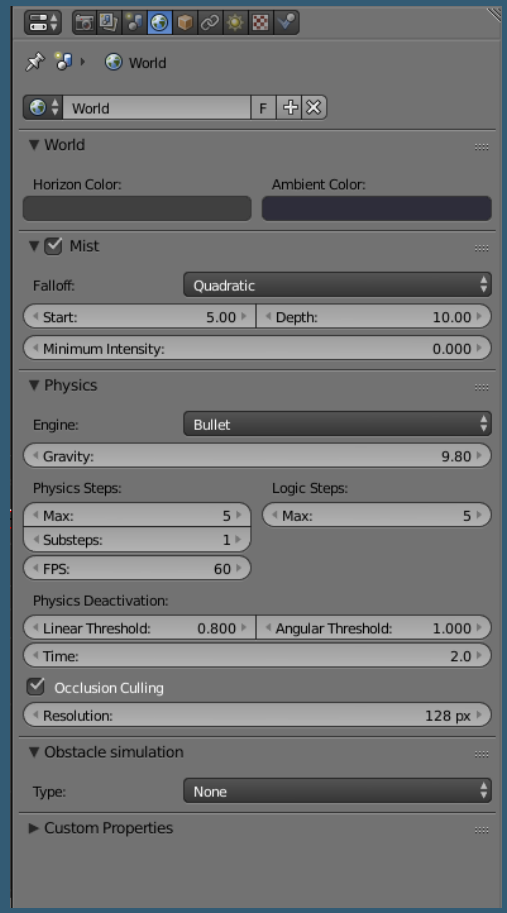
- Skalierter Cube als Spielfeld
- Icosphere als Murmel
- Sun als Beleuchtung





WORLD

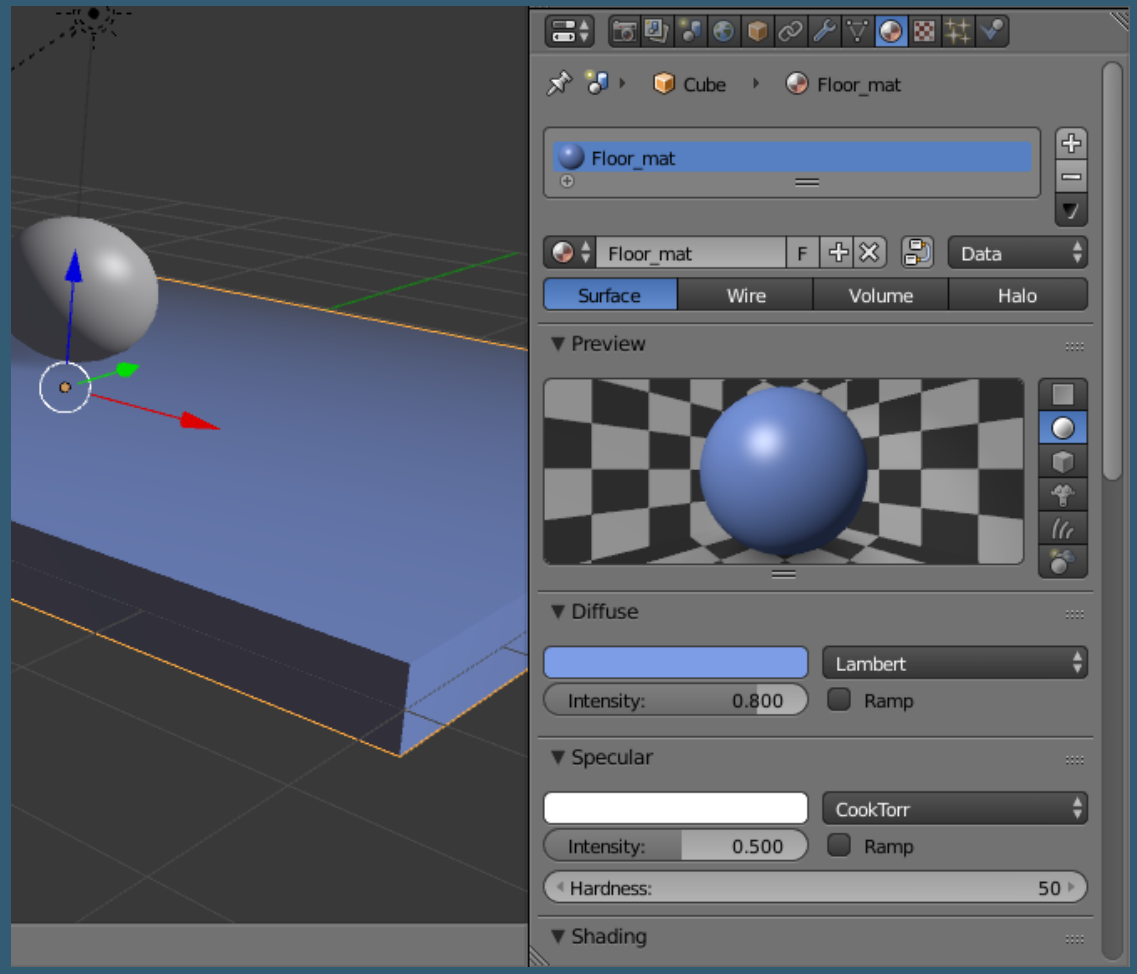
- Properties Panel → World
  - Horizon Color: Nebelfarbe
  - Ambient Color: Ambientes Licht
  - Mist: Nebel
- Änderungen werden erst nach Bewegung in der 3D-View umgesetzt!



Properties Panel

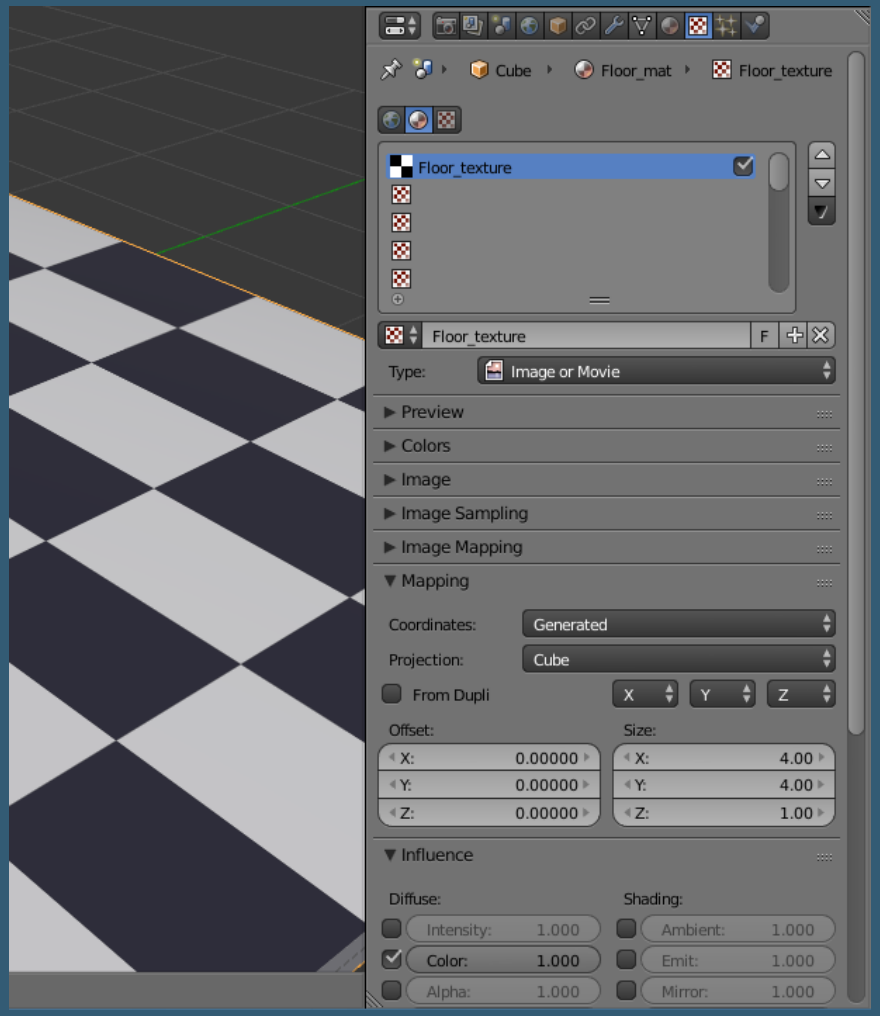
MATERIALIEN (MIT MENÜS)

- Spielfeld selektieren
- Properties Panel → Material
- Sollte schon ein Material haben, da aus anfänglichen Cube entstanden
- Diffuse: Ungerichtete Strahlung
- Specular: Gerichtete Strahlung



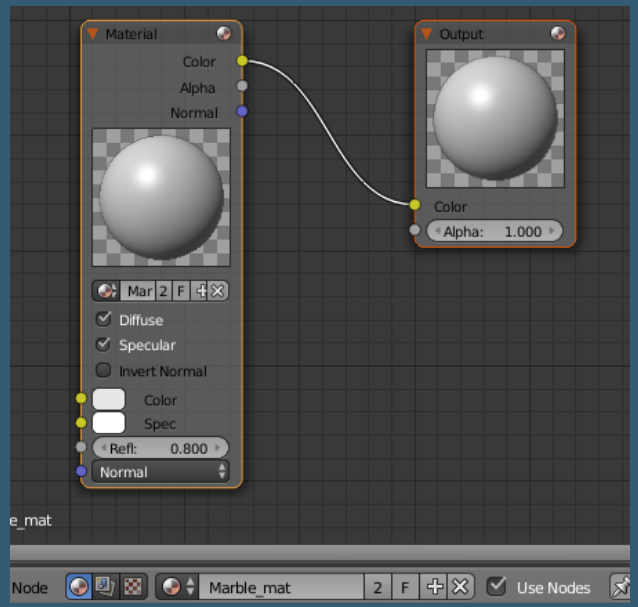
MATERIALIEN (MIT MENÜS)

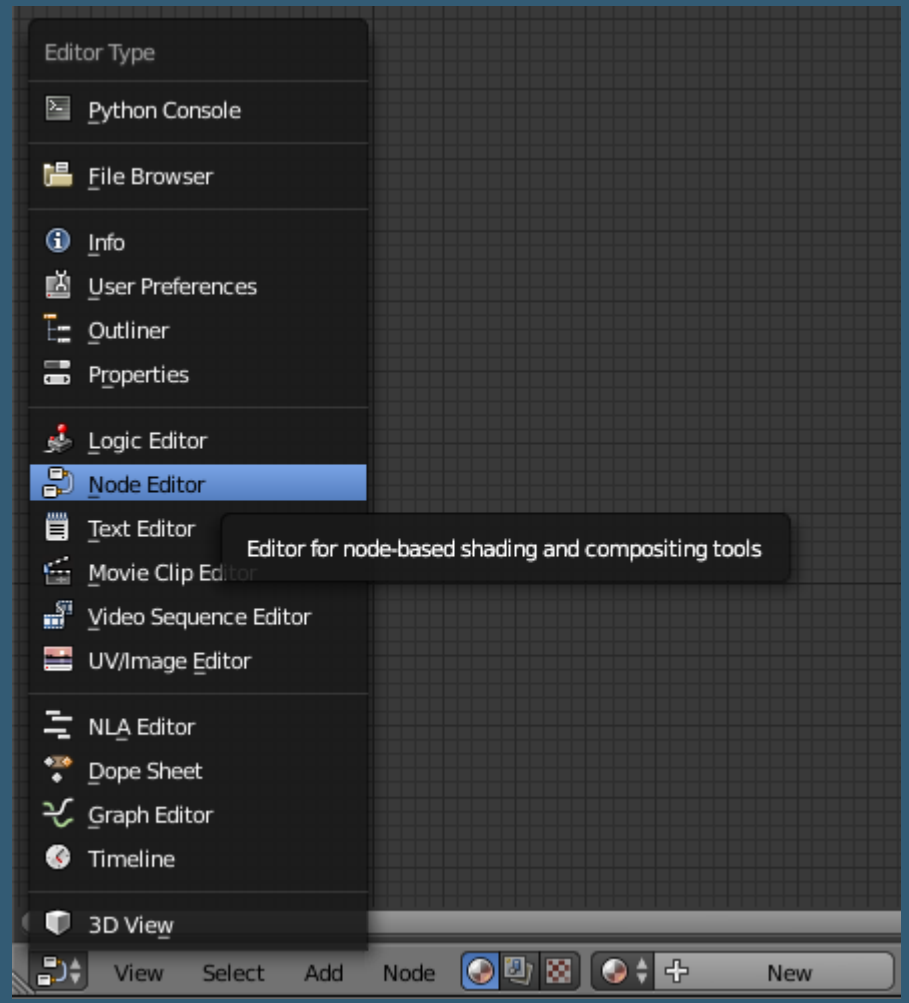
- Properties Panel → Texture
- Es sollte eine leere Textur schon vorhanden sein
- Type: Art der Textur, es funktionieren in der Blender Game anscheinend nur Image (und vielleicht Movie / Environment Map?)
- Mapping: Projektion des Bildes auf das Objekt
- Influence: Einfluss der Textur auf die Materialeigenschaften



MATERIALIEN (MIT NODES)

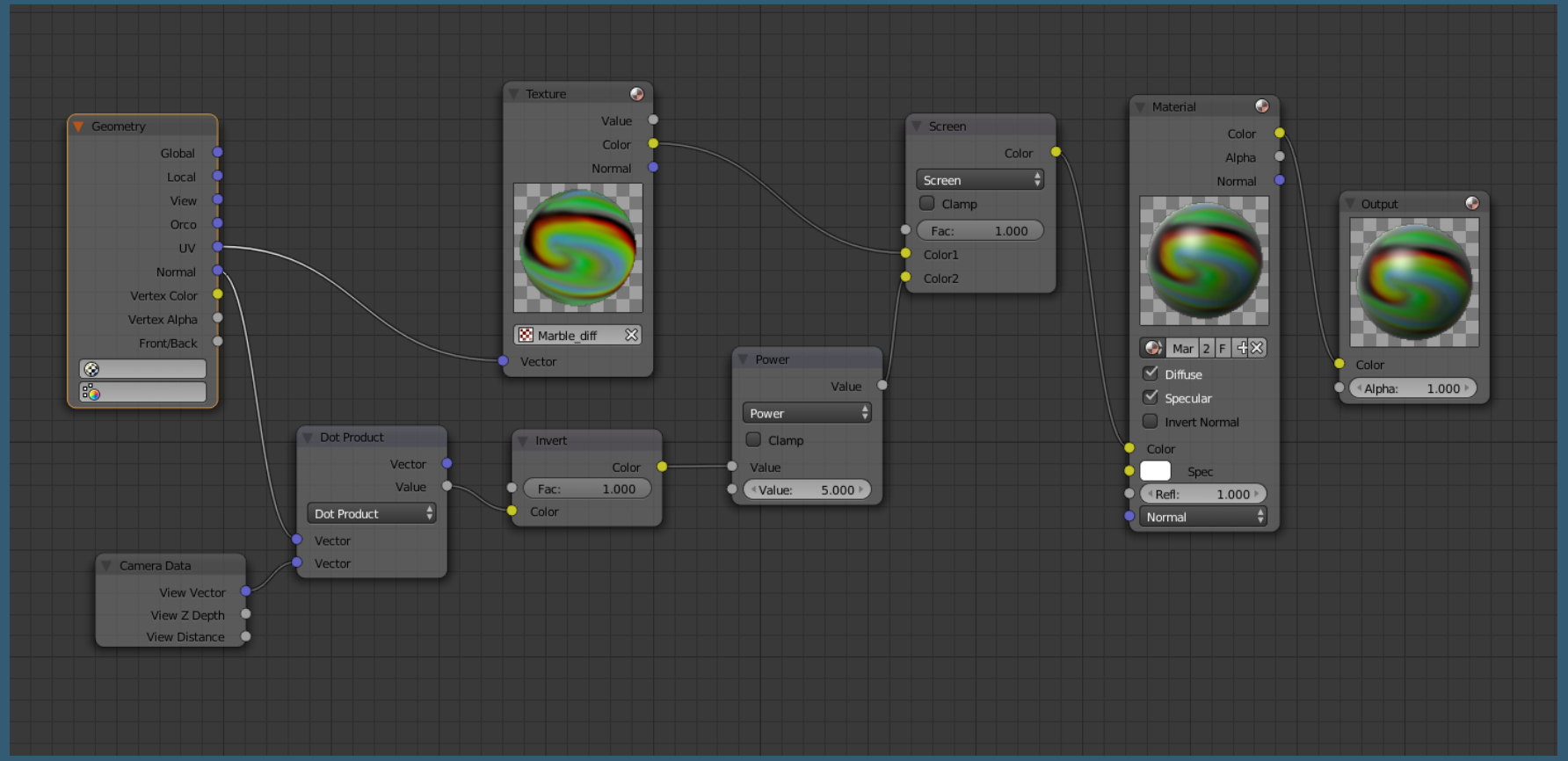
- Murmel selektieren
- Properties Panel → Material
- Node Editor in einem Viewport öffnen
- Neues Material mit Button unten in der Leiste erstellen
- Use Nodes selektieren
- Material im Materialnode auswählen





MATERIALIEN (MIT NODES)

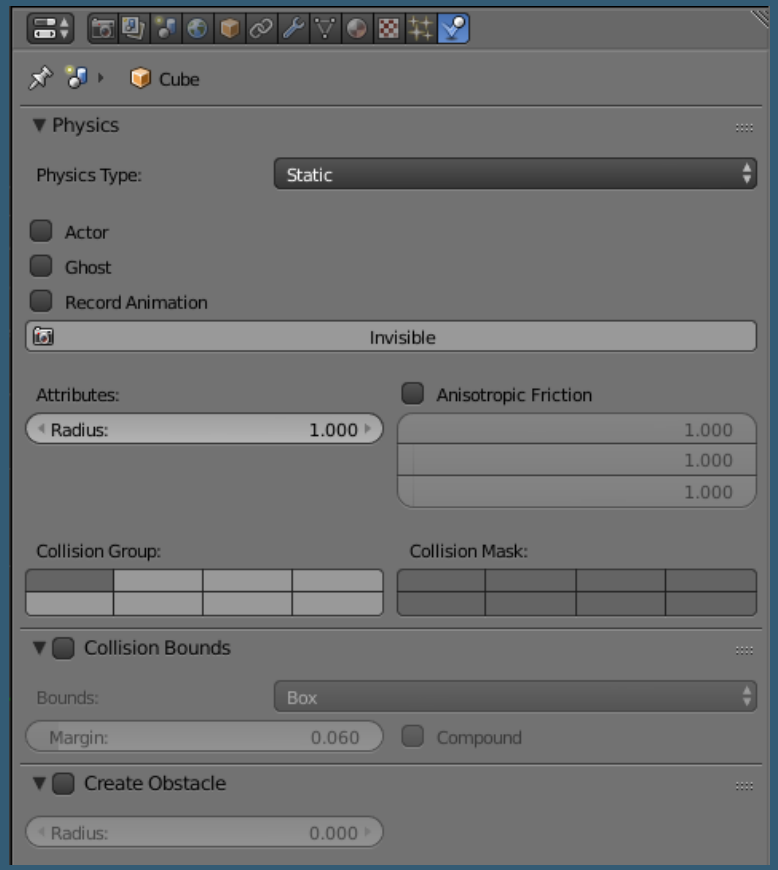
- Setup für Material mit Fresnel



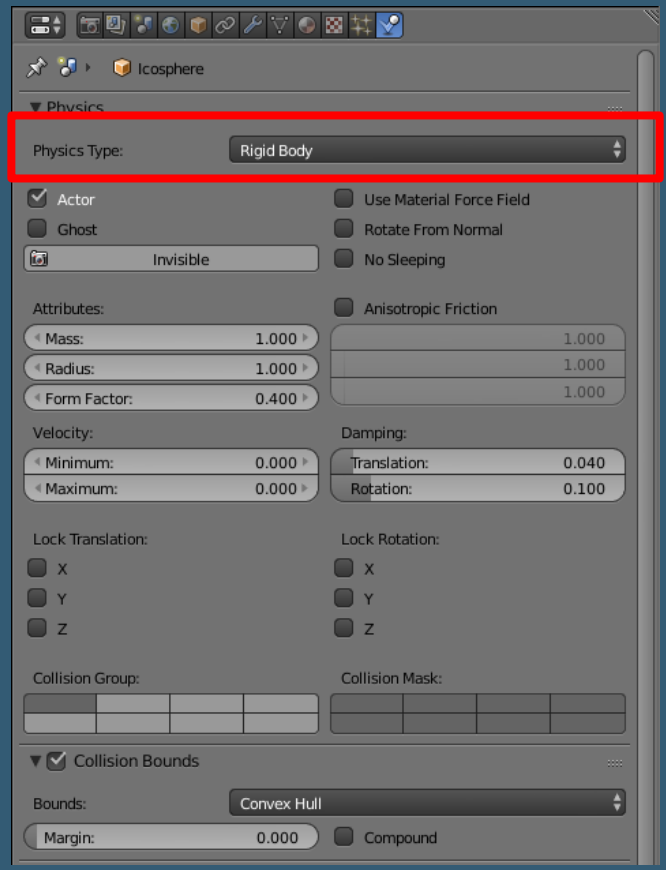
MATERIALIEN (MIT NODES)

- Manches scheint in Blender Game nicht via Nodes zu funktionieren. Diese Einstellungen sind aber weiterhin über das normale Menü erreichbar
  - Shading Formel (Phong, Lambert…)
  - Specular Hardness
  - Emission
  - Physics (Friction usw.)
  - Transparenz
- Auch Einflüsse von Texturen können oder müssen über das alte Menü gemacht werden (dafür muss im Node Editor der Material Node ausgewählt sein)
  - Normal Map
  - Emissive Map

PHYSIK



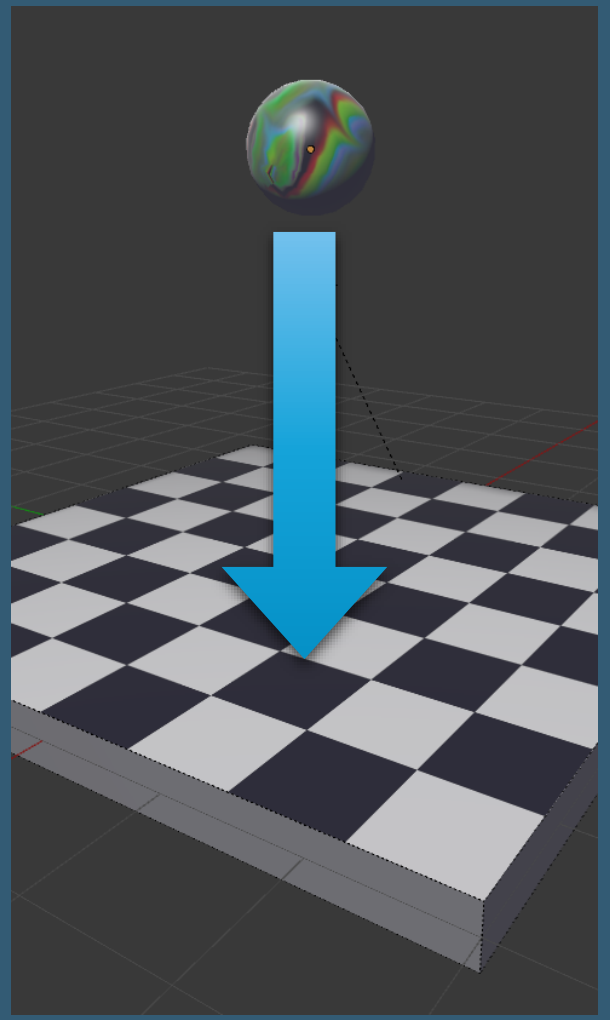
Spielfeld (alles Standard)



Murmel

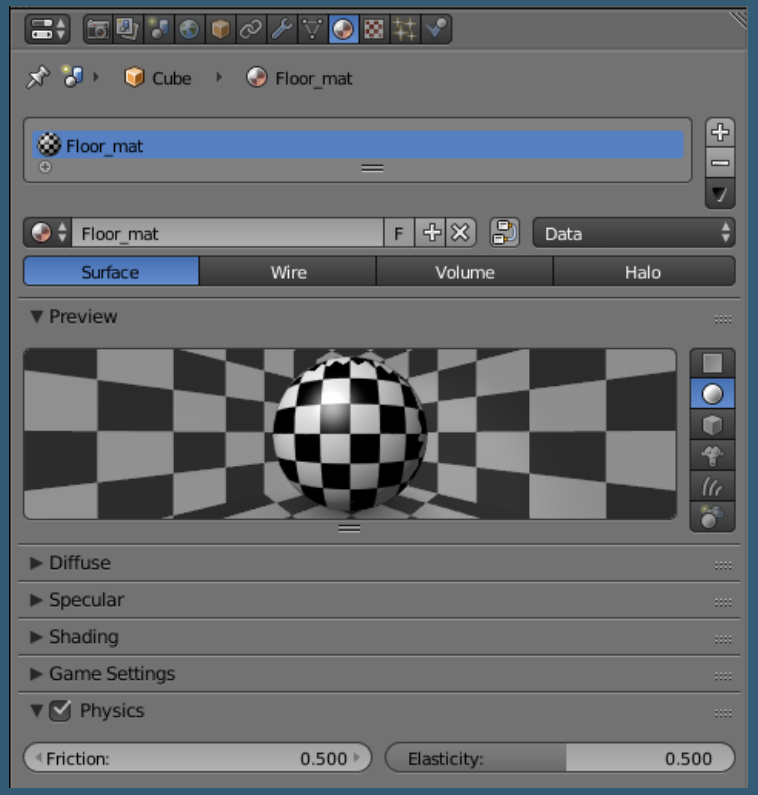
PHYSIK

- „P“ in 3D-View drücken, um das Spiel zu starten
- Murmel sollte auf das Spielfeld fallen
- Sie springt aber nicht hoch sondern bleibt einfach liegen

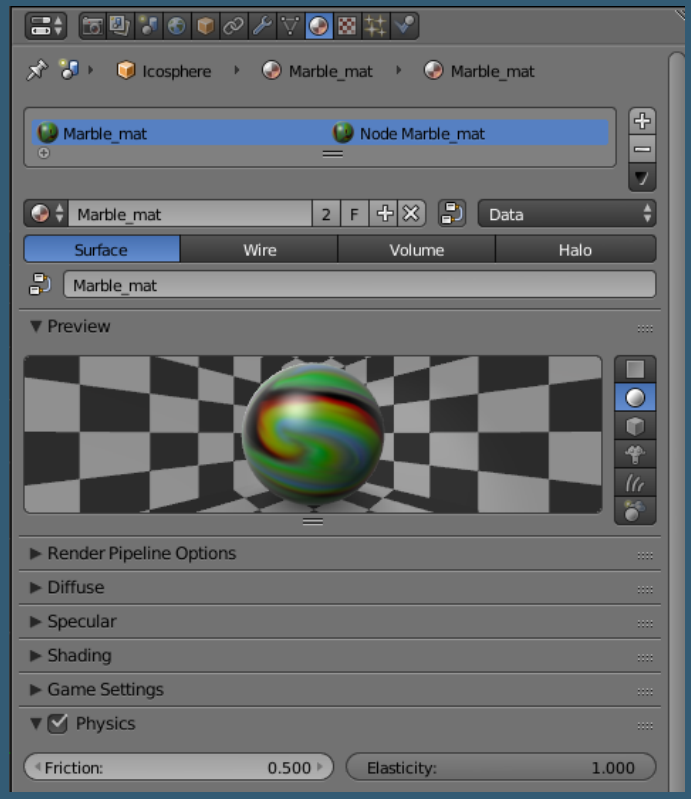


PHYSIK EIGENSCHAFTEN

- In den Materialeigenschaften kann man auch die physikalischen Eigenschaften festlegen
  - Friction: Reibung
  - Elasticity: Elastizität
- Werte zweier kollidierender Materialien werden jeweils verrechnet (wahrscheinlich multipliziert, hängt von Bullet Physics ab)



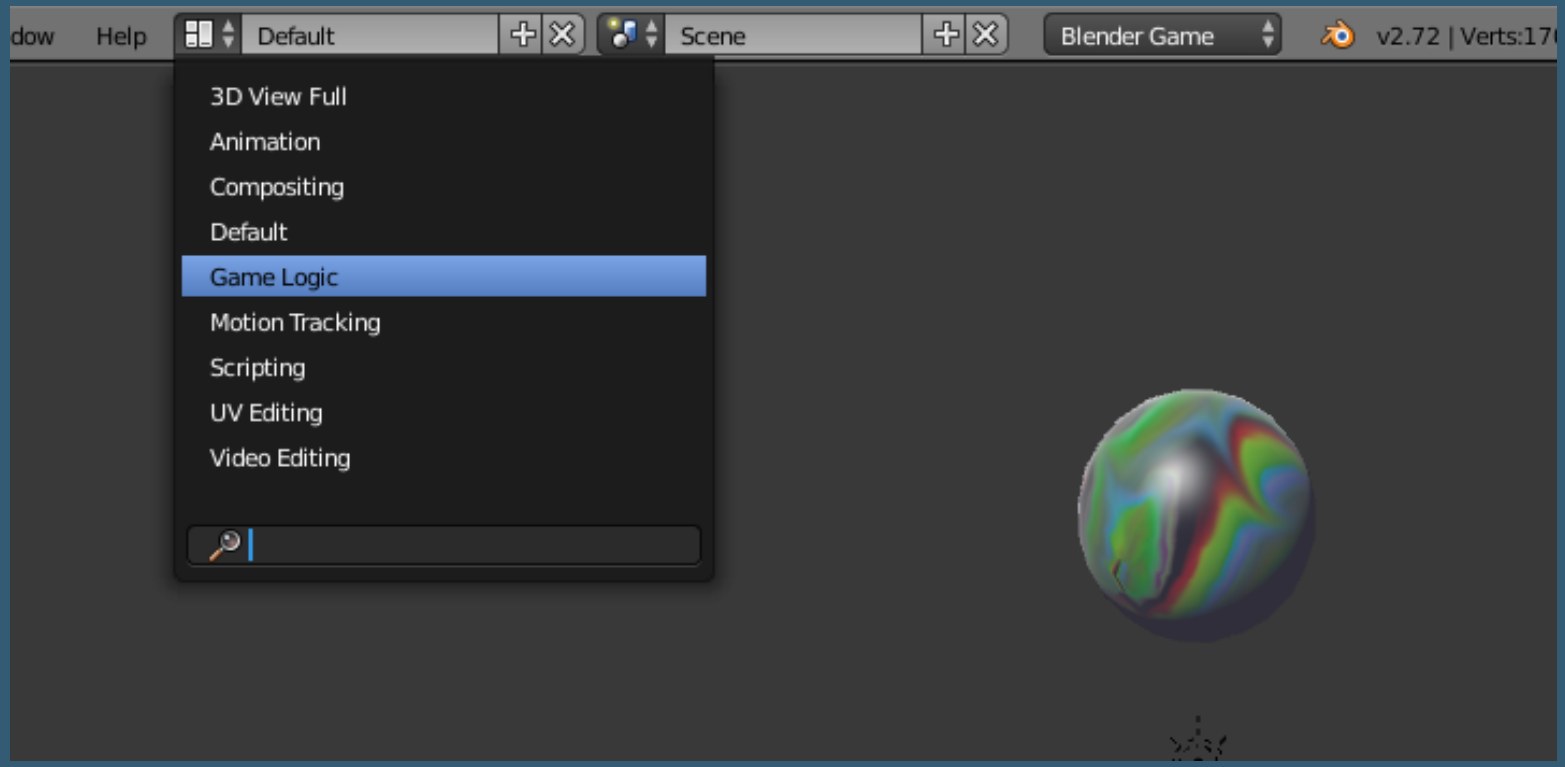
Spielfeld



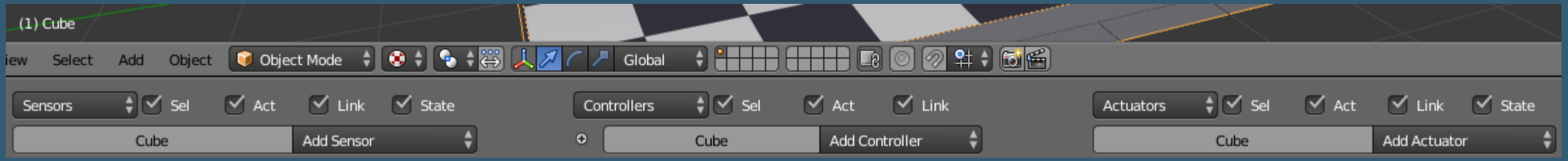
Murmel

GAMEPLAY SETUP

- Spielablauf wird über Logikblöcke und Python-Skripte erstellt (heute kein Python)
- Entweder manuell Viewport mit Logic Editor öffnen oder Screen wechseln



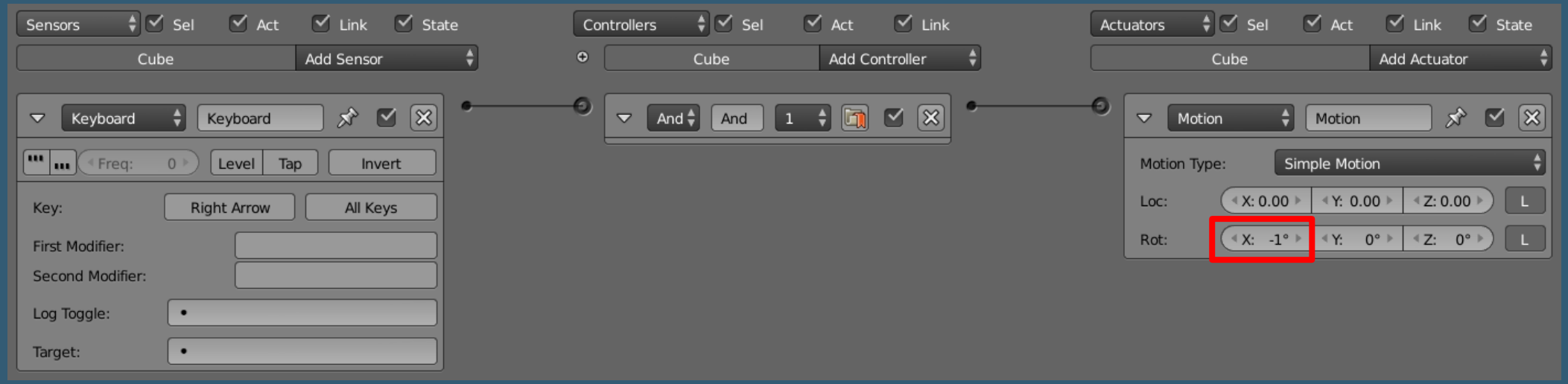
GAMEPLAY LOGIK



- Logikblöcke gehören zu einem Objekt
- Es gibt drei Arten von Blöcken
  - Sensors → Fühlen in die Welt oder werden anderweitig ausgelöst
  - Controllers → Interpretieren die Sensoren mit logischen Operationen oder Python
  - Actuators → Beschreiben (zumeist) Auswirkung auf das Objekt selbst
- Controller können pro Objekt auch noch in States gepackt werden, um Zustände besser zu ordnen (wird hier nicht benutzt)
- Globale Skripte usw. werden oft an das Kameraobjekt gehängt

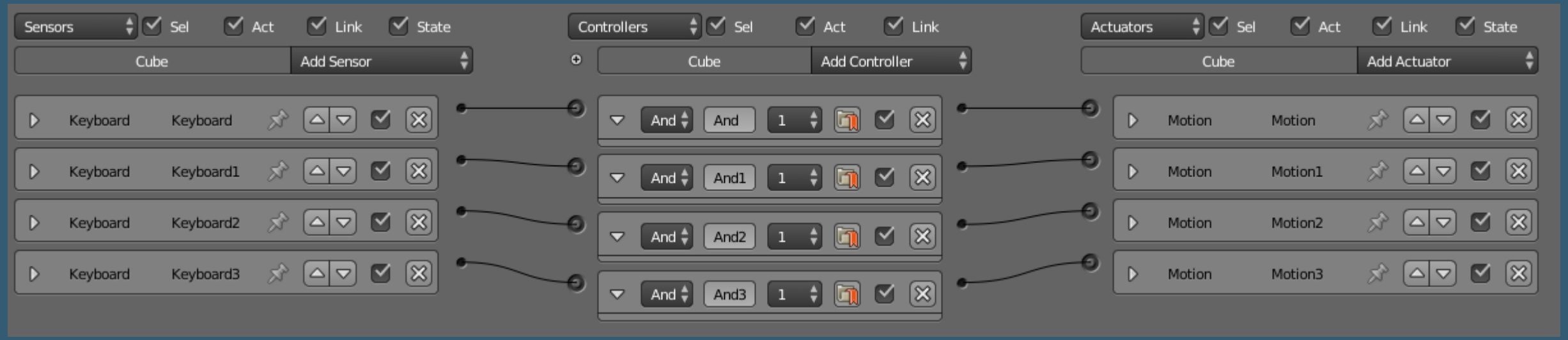
GAMEPLAY LOGIK

- In unserem Fall soll sich das Spielfeld durch Tastendruck neigen lassen
- Daher benötigen wir einen (bzw. später vier) Sensoren, die Tastaturanschläge erkennen
- Einen simplen Controller, der bei Tastenanschlag einen Actuator aufruft
- Einen Actuator, der das Spielfeld neigt
- Die Blöcke werden durch linke Maustaste miteinander verbunden



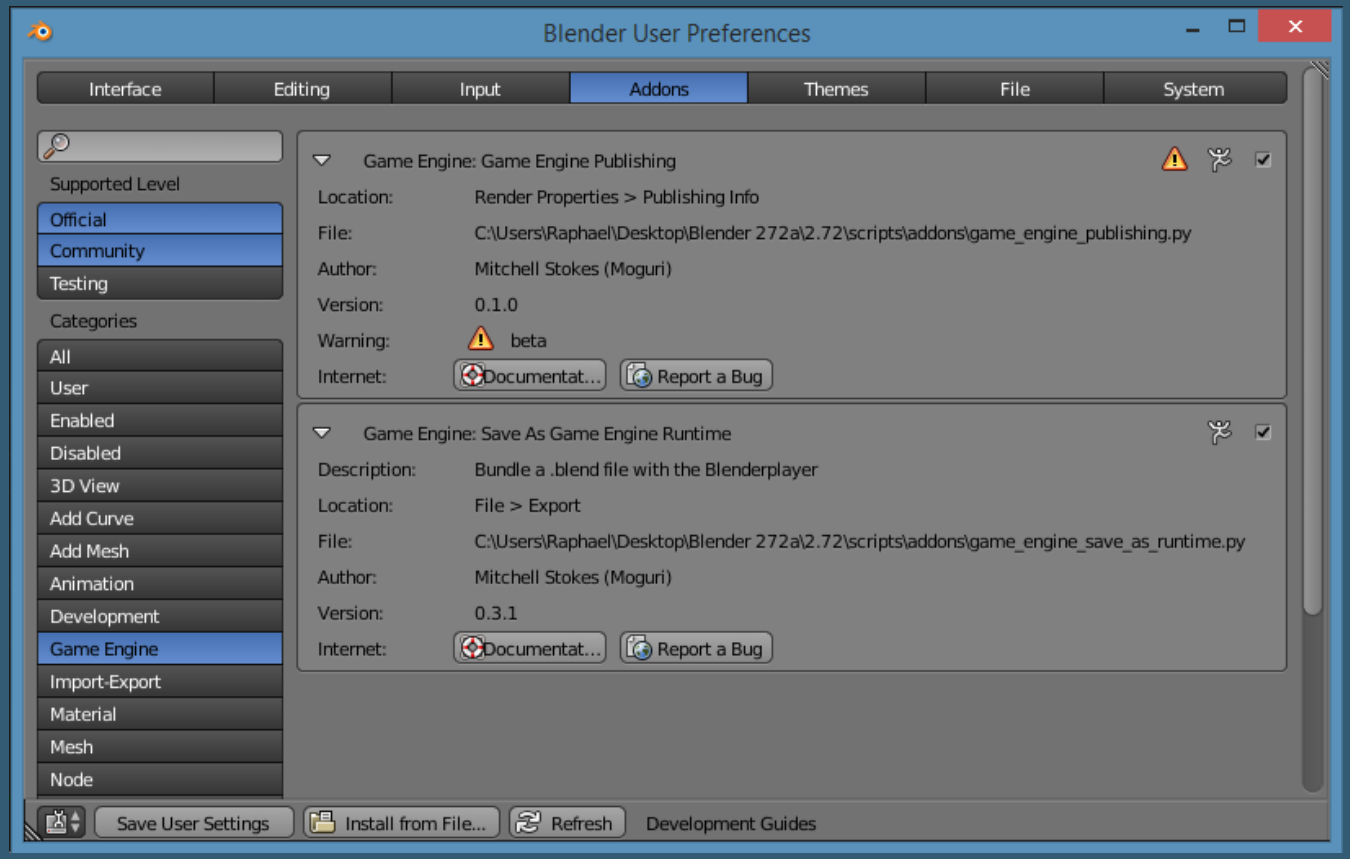
GAMEPLAY LOGIK

- Da sich das Spielfeld in alle vier Richtungen drehen lassen soll, sind mehr Blöcke notwendig…
- Geschaffene Verbindungen lassen sich mit STRG + Linker Maustaste wieder zerschneiden

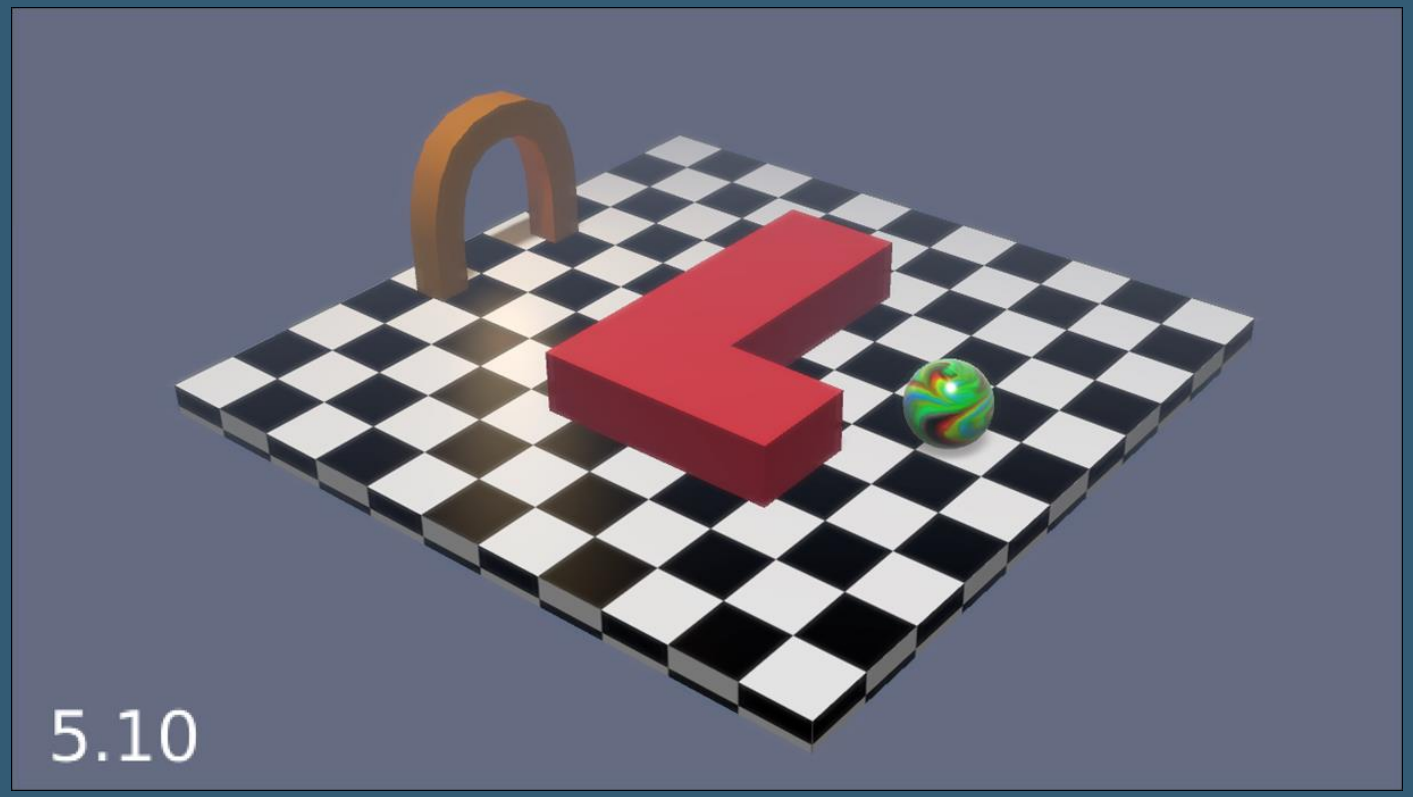


DEPLOYMENT

- Standardmäßig schaut man durch die Hauptkamera der Scene
  - Sollte man das in Blender schon testen wollen, so hilft Numpad + 0
  - Falls jemand mehr erfahren möchte → Google / Bing / Duckduckgo
- Für das Deployment kann man ein Blender Addon benutzen
- Blender Kopfleiste → File → User Preferences
- Es gibt zwei Addons (das untere funktioniert bei mir nur manchmal…)
- Relativ gelinkte Assets wie Bilder können fehlen
- Deployed nur für das aktive Betriebssystem



ZIEL ERREICHT? – NICHT GANZ



Nehmen und anzeigen der Zeit, Logik für Gewinnen oder Verlieren des Spiels…
Schaut hierfür ins fertige Spiel oder kommt in die nächste Session !
(dort werden wir anhand eines Jump‘n‘Run etwas Python machen)