# Die Simulation:

## Aufgabe:

Entwicklung einer Simulation mit folgenden Vorraussetzungen:

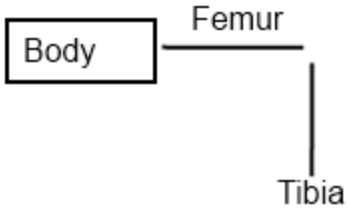
- 3 Zufallsparameter die auf das System Einfluss haben
- 4 beeinflussbare Parameter
- 1 Zufallsparameter sollte min. 1 beeinflussbaren Parameter beeinflussen
- 1 beeinflussbarer sollten einen anderen beeinflussbaren beeinflussen

## Ziel:

Ziel der Simulation ist es verschiedene Bewegungsabläufe des Roboters zu testen und welchen Einfluss u.a. das Gewicht, die Umgebung oder die Beschaffenheit der Beine darauf hat.

## Wichtige Begriffe:

In der Welt der Roboter bedient man sich oftmals der Begriffe aus der Welt der Insekten um den Aufbau des Körpers zu beschreiben. Da unser Roboter sehr simpel gehalten ist gibt es nur 3 Bestandteile, der Body, Femur und Tibia. Um zusätzliche DOF(Degrees of Freedom) zu bekommen, könnte man ein weiteres Gelenk hinzufügen. Dieses würde dann der Tarsus sein.




Mehr Information: http://en.wikipedia.org/wiki/Arthropod_leg

## Umsetzung:

Zur Umsetzung des Projektes wurde nach einer langen Evaluationsphase eine Kombination aus VPython für die grafische Darstellung, und pyODE , Python Wrapper für die in C/C++ geschrieben Physikengine, genommen.

OpenGL hätte auch zur Umsetzung genommen werden könnte, aber VPython hat eine simplere API und dadurch erzielt man schneller seine gewünschten Ergebnisse. Das gleiche gilt warum die Bullet Physics Engine nicht genommen wurde. Diese ist zwar sehr gut und bietet mehr Möglichkeiten als ODE, jedoch wäre es für unser Projekt einfach zu viel.

Viele Toolkits die in Erwägung gezogen worden findet man hier und hier. Viele waren zu komplex, schlecht portierbar(ROS läuft nur auf UBUNTU nach einer komplizierten Installation), keine geeigneten Sprachkentnisse (Java Toolkits) oder waren nicht für meine Anforderungen geeignet (Eher Simulation von z.B. Fahrzeugen oder nur Armen).

# Applikationshinweise:

Es wird eine Auflösung von min. 1280x800 benötigt. Innerhalb der Welt kann man mit der Maus navigieren. Rechte Maustaste verändert die Position der Kamera und dreht diese. Linke+Rechte Maustaste ist für das Zoomen.

Falls der Python Code ausgeführt werde soll, benötigt man folgende Pakete:

- Python 2.6
- VPython
- pyODE
- visualPyOde(mit im Verzeichnis da die Version aus dem Internet durch mich verändert wurde und nicht mehr kompatibel ist)

# Zufallswerte:

Zufällig sind:

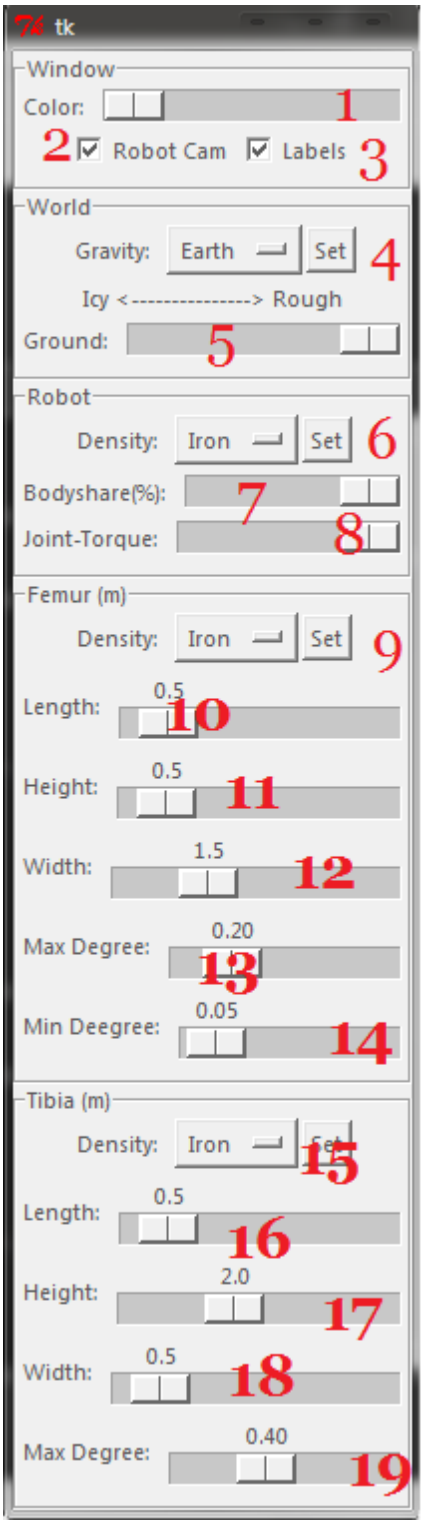
1. Die Anzahl der zusätzlichen Gegenstände(Hindernisse)
2. Die Größe dieser Gegenstände(Breite, Höhe, Länge)
3. Die Masse der Gegenstände
4. (Die Farbe)
5. Die Position der Einzelnen Gegenstände, ist jedoch abhängig von der derzeitigen Position des Roboters
    a. Dieser Zufallsparameter wird zwar von einem beeinflussbaren Parameter beeinflusst, jedoch wirken dadurch die Gegenstände erst auf den Roboter und seine Bewegung ein, egal wo er sich gerade befindet. Quasi eine bidirektionale Beeinflussung.

# Kontroll-Elemente (beeinflussbare Werte)

1. Geschwindigkeit zwischen Einzelbewegungen
2. Roboter wird vom Mittelpunkt weggeschossen
3. Manche Änderungen brauchen einen Neu-Initialisierung des Roboters
4. Ausführung der Bewegung 1
5. Ausführung der Bewegung 2
6. Ausführung der Bewegung 3
7. Ausführung der Bewegung 4


**Fett = neu Initialisierung des Roboters durch den Restart Button nicht erforderlich**

**1. Hintergrundfarbe ändern**
**2. Kamera auf den Roboter fixieren**
**3. Die Beschriftung des Roboters ein/ausblenden**
**4. Anziehungskraft ändern**
**5. Untergrundbeschaffenheit ändern**
6. Material des Roboters ändern
7. Anteil des Materials am Körper ändern
**8. Drehkraft der Gelenke ändern**
9. Material der Femur ändern
10. Länge der Femur ändern
11. Höhe der Femur ändern
12. Breite der Femur ändern
13. Max. Winkel den die Femur einnehmen können
14. Min. Winkel den die Femur einnehmen können
15. Material der Tibia ändern
16. Länge der Tibia ändern
17. Höhe der Tibia ändern
18. Breite der Tibia ändern
19. Max. Winkel den die Tibia einnehmen können




Notiz: die meisten beeinflussbaren Werte beeinflussen unseren “globalen” Wert, die Position und Geschwindigkeit des Roboters.

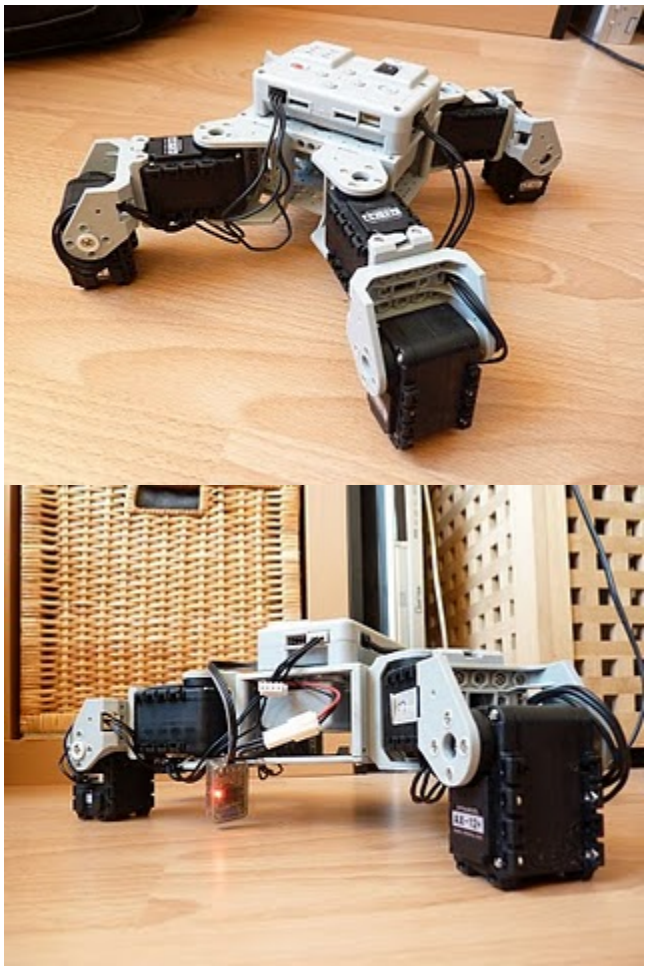
# Zukünftige Pläne:

Bereits geplant wurde das hinzufügen einer Landschaft die durch den Mid-Point Displacement Algorithmus erzeugt wird. Das Programm midpoint.exe zeigt wie solche eine Landschaft aussehen könnte. Die grafische Umsetzung ist relativ einfach, jedoch nicht die Portierung in einen physikalischen Körper der Kollisionen erkennt. Zudem ist der Speicherbedarf dieses komplexen Körpers recht hoch, daher wurde es nicht in diese Version implementiert.

Tip: Es empfiehlt sich die midpoint.exe mehrfach auszuführen, um zu sehen wie verschieden die Landschaften sein könnten.

# Umsetzung der Ergebnisse:

Bei der Simulation stellte ich fest das kurze Beine wesentlich zur Stabilität des Quadruped beitragen. Das folgende Bild zeigt die Umsetzung des grafischen Roboters in einen Realen.



Der Roboter wurde mithilfe des modular aufgebauten Roboter Kit von der Firma Robotis gebaut.

Dieses erlaubt es sehr flexibel und schnell unterschiedliche Typen von Robotern zu bauen. Das sich in dem Ordner befindliche Video “quadruped walking.flv” zeigt die genaue Umsetzung, die jedoch noch weiter optimiert wurde.