UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Robotic Games

Sommersemester 2020

holger.dieterich@ziti.uni-heidelberg.de

Abschlussprojekte

vom 17.06.2020

Lernziele: Anwendung der Inhalte aus der Vorlesung

Präsentation am 29.07.2020

Abgabe Folien + Code bis zum 28.07.2020 um 23:59 in Moodle.

Allgemeines

Es gibt drei verschiedene Spiele names «cat mouse cheese», «gold rush»und «evader surveillance». Pro Spiel wird in Moodle ein package mit dem entsprechenden Namen zur Verfügung gestellt. Zu jedem Package gibt es derzeit jeweils ein gleichnamiges launch-File. Die launch-Files werden noch um weitere Beispiele erweitert, um die Szenarien an die Lösungsansätze anzupassen. Ein solches launch-File wird später bei einem Spiel gestartet.
Pro Gruppe ist hier ein einziges launch-File vorgesehen, das alle autonomen Funktionen starten muss. Jede Gruppe erzeugt ein (Meta-)Package, das nach dem jeweiligen Team-Nummer benannt sein wird. Bspw. Gruppe 1 erzeugt Package *gruppe1*. Dort ist ein launch file enthalten, names *gruppe1.launch*. So können später die Gruppen einfacher gegeneinander antreten. Im zur Verfügung gestellten Code werden neben den bisher verwendeten Packages (turtlebot3 Siimulation etc.) auch einige andere verwendet, die ihr gegebenenfalls nachinstallieren müsst.

```
sudo apt install ros-melodic-timed-roslaunch
sudo apt install ros-melodic-teleop-twist-keyboard
```

Hinweis: Die zur Verfügung gestellten Packages sind nicht perfekt. Anregungen und Feedback sind wie bisher gern gesehen. Es ist sehr wahrscheinlich, dass Parameter (wie bspw. die maximale Translations-Geschwindigkeit der Roboter) im Laufe der Projektphase angepasst werden müssen, damit die Spiele fair sind. Bitte habt Verständnis hierfür. Wir werden Änderungen immer in Moodle bekannt geben und mittwochs in den Gruppen-Meetings ansprechen.
Organisatorisches:Jeden Mittwoch sind Gruppen-Tutorien vorgesehen. Hier kann und soll jede Gruppe einzeln hauptsächlich um Theorie und Diskussun von Ideen zur Umsetzung Damit wird die Es gibt neben den Mittwochs-Terminen Für die Zeit zwischen 14 und 17 Uhr wird es für jede der zwölf Gruppen einen Zeitslot a 15 Minuten geben. Um diese Zeit möglichst effektiv zu nutzen, ist eine Vorbereitung (in Form von Fragen, Skizzen, etc.) sinnvoll. Der aktuelle Spielstand wird immer unter dem Topic */game_state* angegeben. Im Package *cat_mouse_cheese* existiert das *cat_mouse_cheese.launch*. Um ein Szenario zu erkennen (bspw. zur Auswahl einer Karte), kann der ROS-Parameter *szenario* ausgelesen werden (rosparam get).

1) Cat mouse cheese

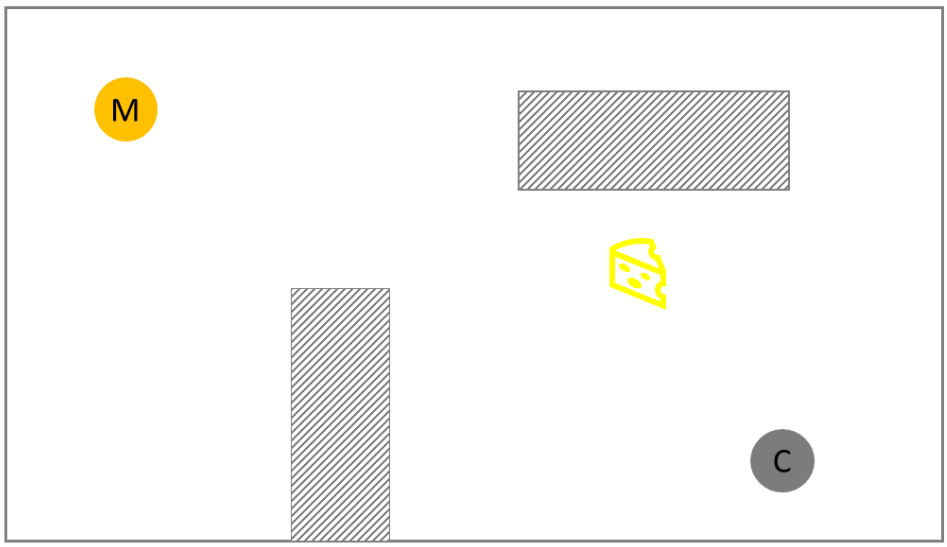

Abbildung 1: Szenario: «cat mouse cheese»

Im diesem Spiel dreht es sich (ähnlich wie in Aufgabenblatt 6) um ein Spiel, in dem einer der Spieler versucht den anderen zu fangen. Dieser versucht natürlich zu entkommen («pursuit-evasion»). Im Gegensatz zu Aufgabenblatt 6 ist das Ziel der Maus hier nicht nur das reine Überleben: Die Maus möchte sich den Käse zu schnappen ohne dabei gefangen zu werden.

Die vorgegebene Lineargeschwindigkeit v_x in x-Richtung soll für Katze und Maus durchgängig konstant sein. Katze und Maus sollen also nie stehen bleiben. Die Rotationsgeschwindigkeit wird von den Teams bestimmt und soll in den unten angegebenen Werte für Katze bzw. Maus liegen.

	v_x	ω_z
Katze	0.22 m/s	-0.8 ... 0.8 rad/s
Maus	0.18 m/s	-2.84 ... 2.84 rad/s

2) Goldrush

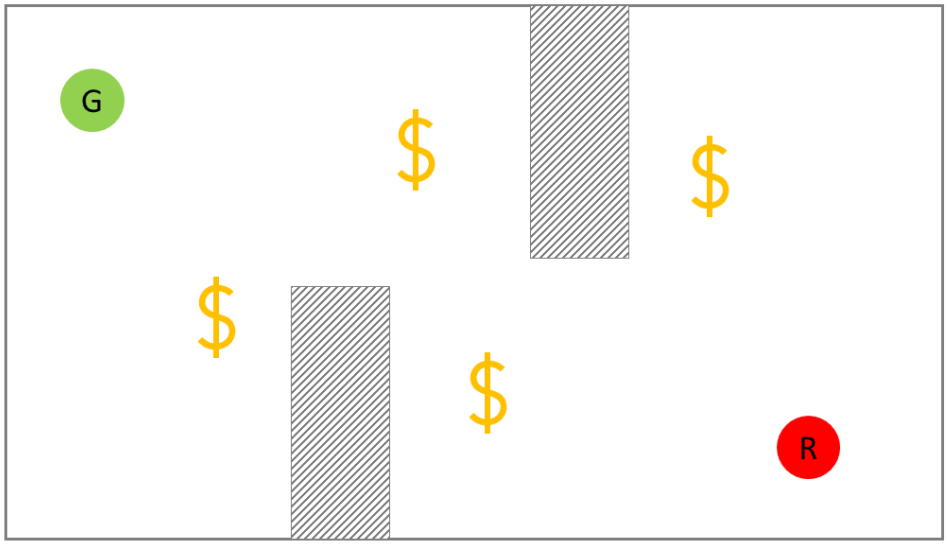

Abbildung 2: Szenario: «goldrush»

Bei diesem Spiel liegen in einer Welt verschiedene Goldstücke verteilt. Die einzelnen Stücke haben unterschiedliche Werte (siehe Topic *golds*). Gold-Stücke können aufgesammelt werden, indem ihre ungefähre Position angefahren wird. Das Spiel ist zu Ende, wenn alle Gold-Stücke aufgesammelt sind. Gewonnen hat der Spieler, dessen Summe aller Gold-Werte dann am größten ist.
Das Launch-File einer Gruppe soll mit einem Parameter aufrufbar sein, das die jeweilige Farbe der Gruppe bestimmt (*siehe goldrush.launch*).

Die maximal zulässigen Soll-Werte für Translations- und Rotationsgeschwindigkeit für rot und grün sind:

	v_x	ω_z
rot/grün	0.22 m/s	-2.84 ... 2.84 rad/s

3) Evader Surveillance

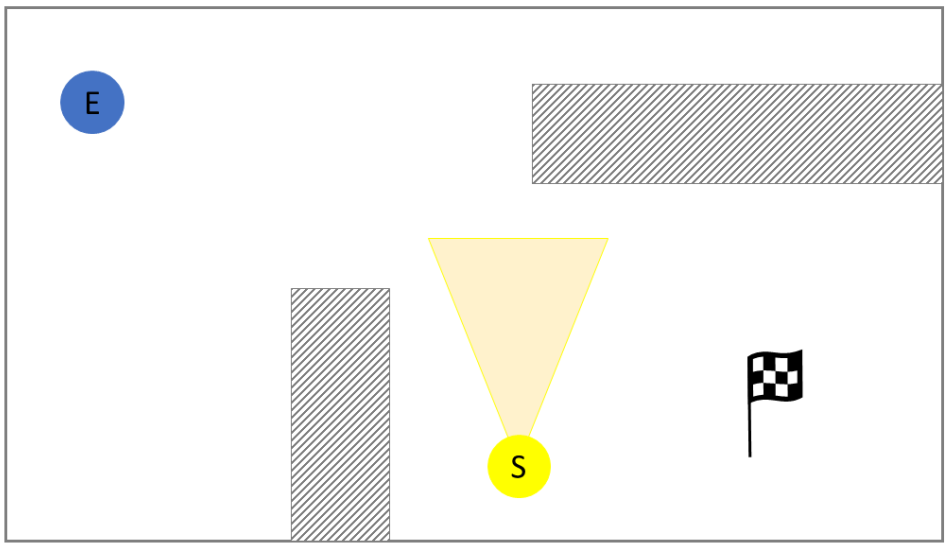

Abbildung 3: Szenario: «evader surveillance»

In diesem Spiel wird ein zu schützender Bereich (Position + Radius, siehe Code) von einem Wärter bewacht. Der Wärter hat ein beschränktes Sichtfeld in Form eines Dreiecks. Ein Eindringling versucht (ohne vom Wärter gesehen zu werden) den Ort zu erreichen. Schafft dies der Eindringling, hat er gewonnen. Erreicht der Eindringling die Position nicht innerhalb von 60 Sekunden oder wird vom Wärter gesehen, dann verliert der Eindringling bzw. der Wärter gewinnt.

Die vorgegebene Lineargeschwindigkeit v_x in x-Richtung soll für Evader und Surveillance durchgängig konstant sein. Evader und Surveillance sollen also nie stehen bleiben. Die Rotationsgeschwindigkeit wird von den Teams bestimmt und soll in den unten angegebenen Werte für Evader bzw. Surveillance liegen.

	v_x	ω_z
Surveillance	0.14 m/s	-1.84 ... 1.84 rad/s
Evader	0.22 m/s	-0.8 ... 0.8 rad/s