Ludwig Maximilians Universität München
Department „Institut für Informatik“
Lehr- und Forschungseinheit für Datenbanksysteme

Skript zur Vorlesung

Managing and Mining Multiplayer Online Games

im Sommersemester 2012

Kapitel 5: Künstliche Intelligenz in Spielen

http://www.dbs.informatik.uni-muenchen.de/cms/VO_Managing_Massive_Multiplayer_Online_Games

Kapitelübersicht

- Steuerung von mobilen Objekten
- Regelbasiertes Verhalten
- Zustandsautomaten
- nicht deterministisches Verhalten
- Algorithmen für AI-Engines
  - Wegewahl in offenen Umgebungen
  - Antagonistisches Verhalten

Aktionserzeugung für NPCs

NPCs (Non-Player Characters), **MOBs** (Mobile Objects) oder **Bots** (automatische Steuerung von Spielercharakteren) müssen Aktionen erzeugen um handeln zu können.

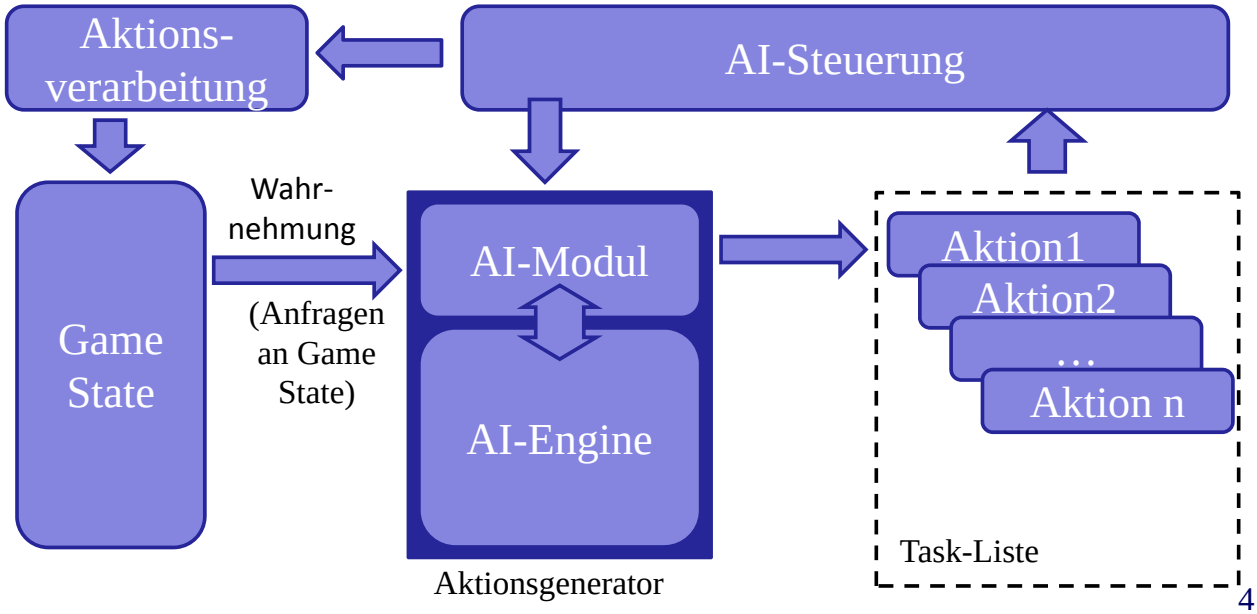
wichtige Aspekte:

- Granularität: Steuerung einer oder mehrerer Game Entities
- Berechnungsaufwand vs. Verhaltenskomplexität
- Wo werden NPCs berechnet (Server/Client-seitig)
- deterministisches vs. probabilistisches Verhalten
- Realitätsnähe des Verhaltens

Grundaufbau eine KI

Aufgabe:

- Generiere eine/mehrere Handlungen
- die dem jetzigen Zustand angemessen sind



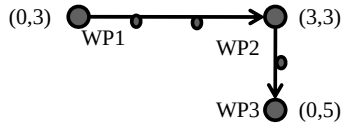
AI-Steuerung

Synchronisation von Aktionsverarbeitung und Aktionserzeugung:

- Aufruf der Aktionserzeugung
  - **Zeitgesteuert**: Handlungswünsche werden regelmäßig erfragt
  - **Event-gesteuert**: Bei bestimmten Änderungen des GS können Handlungswünsche erfragt werden.
    (Spieler kommt in Aggro-Reichweite eines Mobs)
- Durchführung von Tasklisten
  - **Verarbeitungszeitpunkte** einzelner Aktionen in der Task-Liste
    =>Durchführen komplexer Handlungen (Sequenzen von Aktionen)
  - Aktion die über einen **Zeitraum** ausgeführt werden und Management von Cool Downs (CDs)

Beispiel:

Bewegung entlang einer Trajektorie:
(WP1,WP2,WP3)



Tick	X	Y	State	
1	0	3	Gehe zu WP2	aus Task-Liste
2	1	3	Gehe zu WP2	
3	2	3	Gehe zu WP2	
4	3	3	Gehe zu WP3	aus Task-Liste
5	3	2	Gehe zu WP3	
6	3	1	Gehe zu WP3	

Aktionsgenerierung

- Wann und was wird vom Game State angefragt?
  Eine zentrale Anfrage an Game State („alle Wahrnehmungen“)
  vs. mehrere Anfragen (selektive Wahrnehmungen)
- Wie werden Handlungen generiert?
  - Organisation über Regeln oder Zustandsautomaten
  - Deterministische Handlungen oder zufallsbasiert?
- Welche Möglichkeiten soll es beim Einfügen in die Taskliste geben?
  - Löschen alter Handlungen
  - Einfügen in die Taskliste(Anfang, Ende, ..)
- Umsetzung
  - Programmierung in der Sprache der Game Engine
  - Programmierung über Scriptsprache (z.B. LUA)

Beispiel: „Autocamp 2000“

Beispiel eines Bots: **(http://archive.gamespy.com/fargo/august03/autorpg/)**

1) If invited by any group => join group
2) If in a group => follow behind the leader
3) If sees a monster => attack
4) If someone says something ending in a question mark
   =>respond by saying "Dude?"
5) If someone says something ending in an exclamation point
   => respond by saying "Dude!"
6) If someone says something ending with a period
   => respond by randomly saying one of three things: "Okie," "Sure," or "Right on."
7) EXCEPTION: If someone says something directly to you by mentioning your name, respond by saying "Lag."

Beispiel

KillSwitch: [Shouting] Does anyone want to join our hunting party?

Farglik: [Powered by the Autocamp 2000] Dude?

[KillSwitch invites Farglik to join the group.]

[Farglik joins the group]

KillSwitch: We're gonna go hunt wrixes.

Farglik: Right on.

[The group of players runs out, Farglik following close behind. Farglik shoots at every little monster they pass.]

KillSwitch: Why are you attacking the durneys?

Farglik: Dude?

KillSwitch: The durneys, the little bunny things -- why do you keep shooting at them?

Farglik: Dude?

KillSwitch: Knock it off guys, I see some wrixes up ahead. Let's do this.

Farglik: Right on.

[The group encounters a bunch of dangerous wrixes, but they gang up and shoot every one of them.]

KillSwitch: We rock!

Farglik: Dude!

Troobacca: We so OWNED them!

Farglik: Dude!

Beispiel

KillSwitch: Uh oh, hang on. Up ahead are some Sharnaff bulls. We can't handle them, so don't shoot.
Farglik: Okie.
[Farglik shoots one of the Sharnaff bulls.]
[The bull attacks; Trobacca and several other party members are killed before they beat it.]
KillSwitch: You IDIOT! Farglik why did you shoot at them?
Farglik: Lag.
KillSwitch: Well don't do it again.
Farglik: Sure.
[Farglik shoots at another Sharnaff bull.]
[The entire party is slaughtered except for Farglik.]

[... Farglik stands there, alone, for several hours ...]

Steuerung durch Regeln

- **Rumpf**: Prädikatenlogischer Ausdruck auf dem Game State
- **Kopf**: Liste von Aktionen, die in die Task-liste eingefügt werden
- Steuerung besteht aus einer Liste von Regeln.
- Auswahl der Regel über Sortierung nach Priorität
  => erste Regel die feuert erzeugt Handlung

Nachteile:

- Regelrümpfe sind häufig redundant
- hoher Aufwand beim finden der anzuwendenden Regel

Beispiel:

- NPC.inbattle() $\wedge$ (NPC.health < 50) => NPC.runAwayFrom(NPC.battle().getMaxAggroMob())
- NPC.inbattle() $\wedge$(NPC.health > 50) => hit (NPC.battle().getMaxAggroMob())
- (NPC.health < 50) $\wedge$(NPC.inv.contains(Melone)) => NPC.eat(Melone)
- NPC.inGroup() => Follow(NPC.GroupLeader)
- NPC.hasInvite() => NPC.join(NPC.invitation.getGroup())
- __=> NPC.wait()

Steuerung mit Zustandsautomaten

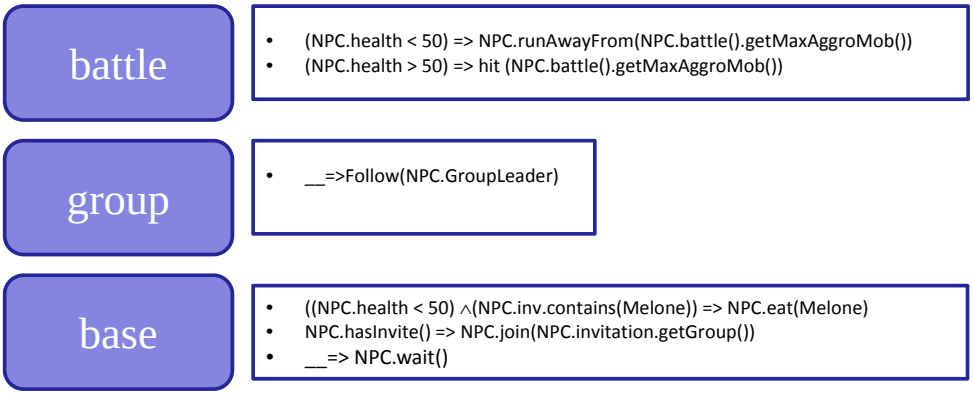
Idee: Fasse regeln nach Zustand des NPCs zusammen.

⇒ nur die Regeln, die beim aktuellen Zustand sind, werden ausgewertet

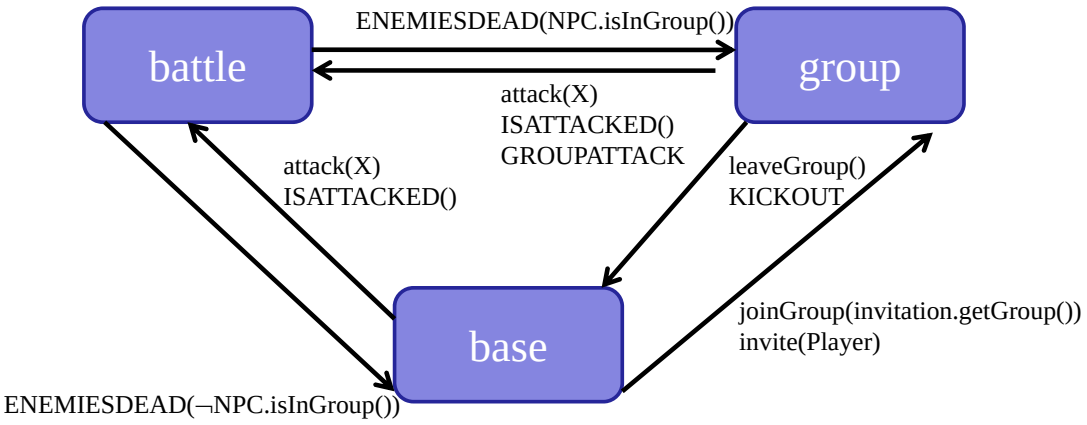
⇒ aktueller Zustand des NPCs ist Teil des Game State

⇒ Zustände können bei komplexen MOBs auch weiter unterteilt werden

⇒ Änderungen des Zustands über Aktionsverarbeitung
(z.B. Monster bricht die Verfolgung ab => Übergang von *battle* zu *base*)



Beispiel: Zustandsautomat



- GROSSBUCHSTABEN: Eventbasierter Übergang
- Kleinbuchstaben: Aktionsbasierter Übergang

Verarbeitung mit Zustandsautomaten

Ablaufschema:

1. Anfrage des ***GS***
   (zustandsbestimmende Informationen)
2. Bestimmen des aktuellen Zustands
   - eindeutige Aufteilung über disjunkte Zustände (z.B. Krieg oder Frieden)
   - eindeutige Bestimmung über Prioritäten(z.B. 1:battle, 2:group, 3:base)
3. Anfrage des ***GS***
   (zustandsspezifische Informationen)
4. Aktionsgenerierung
   (Anwendung der lokalen Regeln)
5. Manipulation der Task-Liste
   - Abbruch der alten Aktion:
     Löschen der TL und Neueinfügen der neuen Handlungen
   - Einfügen der neuen Aktionen am Anfang der Liste
   - Einfügen der neuen Aktionen am Ende der Liste

Determinismus und Verhaltensänderung

- ***Deterministisches Verhalten***:
  MOBs generieren in der gleichen Situation immer die gleichen Aktionen
- ***Probabilistisches Verhalten:***
  - Regeln haben im Kopf eine Menge möglicher Aktionen
    =>Auswahl mittels Zufallsprozess
  - in einer Situation können mehrere Regeln anwendbar sein
    => zufällige Auswahl der Regeln
  - die Bestimmung des aktuellen Zustands oder Unterzustands wird zufällig ausgewählt
- ***Lernendes Verhalten:***
  - spezielle Parameter im Kopf oder Rumpf der Regeln werden auf Basis von Trainingsbeispielen angepasst
    (Verwendung von Klassifikations- und Regressionsmethoden)
    Beispiel: Optimierung der Wahrscheinlichkeitsverteilung verschiedener Angriffe bei bestimmten Gegnern
- ***Evolutionäres Verhalten:***
  - Einfügen neuer Regeln und Zustände in Rahmen der allgemeinen Spielregeln
  - Training über z.B. über genetische Algorithmen
  - Problematisch, da viele neue Regeln erfolglos sein werden

  (sehr experimentell und in den meisten Spielen kaum umsetzbar)