Übersicht: Das Event-System im Insekten-Simulator

12. Mai 1998

1 Events auf der `Sim`-Ebene

Die Events auf Simulationsebene werden in `Sim::Insect` definiert und sind für die Übermittlung von Ereignissen vom Server zum Client zuständig.

Es wird dabei wird zunächst unterschieden zwischen

- `DoEvents` und
- `ObservationEvents`

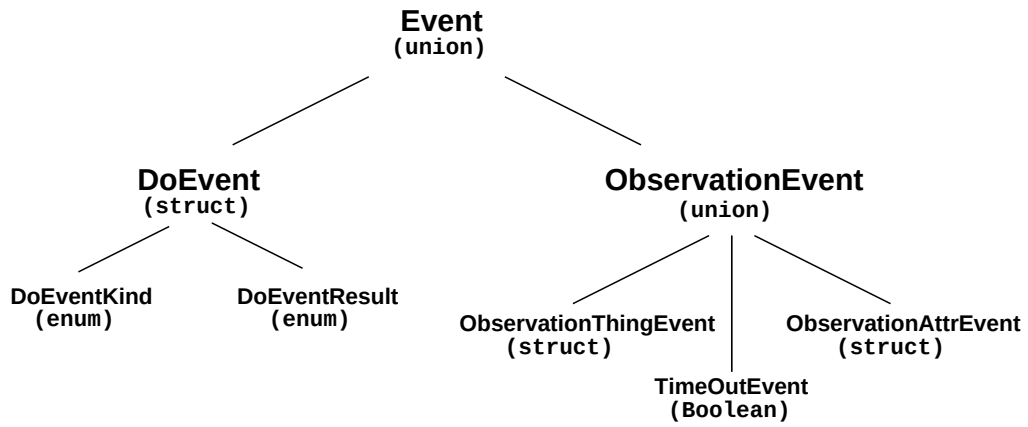

Abbildung 1: Struktur der Events

Erstere sind für Ergebnisse von Aktionen zuständig, werden allerdings von der momentanen Implementation nicht unterstützt, letztere sind für die Übermittlung von Beobachtungen des zugehörigen Insektes zuständig.

Die `ObservationEvents` lassen sich noch einmal unterteilen in

- `ObservationAttrEvents`
- `ObservationThingEvents`
- `TimeoutEvents`

Erstere stellen dabei Beobachtungen am eigenen Objekt und deren Attribute sowie an fremden Objekten dar. Dagegen hat der `TimeoutEvent` (im Code als `dummy` bezeichnet) eine Sonderrolle: Er stellt eine Art Zeitstempel dar, der einmal pro Durchlauf der `start()`-Schleife (durch Aufruf von `Observer::check()`) an die Event-Liste angehängt wird, so daß ein Client z.B. entscheiden kann, nur die Events aus einem bestimmten Zeitraum zu verarbeiten.

Zur besseren Übersicht hier noch die Definitionen der Events in `Sim.idl`:

```
union Event switch (EventType) {
  case aDoEvent:            DoEvent Do;
  case aObservationEvent:   ObservationEvent Observation;
};

enum EventType { aDoEvent, aObservationEvent };

struct DoEvent {
  DoEventKind   theKind;
  DoEventResult theResult;
};

enum DoEventKind { Move, Fight, Eat, Hold };
enum DoEventResult { Done, Failed };

union ObservationEvent switch(ObservationEventKind) {
  case Thing:   ObservationThingEvent Thing;
  case Attr:    ObservationAttrEvent Attr;
  case Timeout: boolean dummy;
};

enum ObservationEventKind { Thing, Attr, Timeout };

struct ObservationThingEvent {
  ThingIndex Thing;
  ThingType  Type;
  Position   Pos;
  Volume     Vol;
};
```

```
struct ObservationAttrEvent {
  AttrType  Attr;
  AttrValue Value;
};
```

2 Events auf der Kernel-Ebene

Die Sim-Events werden wie viele andere Datenstrukturen auf Simulationsebene definiert, ihre Implementation findet jedoch auf der Kernel-Ebene statt: Jedes Insekt (KerInsectImpl) verfügt über eine private Event-Liste vom Typ KerEventList, die eine Wrapper-Klasse für die eigentliche Sim::Insect::EventList darstellt. In diese Liste werden einerseits alle physikalischen Events (diese werden in processMove() von der physikalischen Referenz übernommen) mittels KerEventList::insertObservationEvents() eingetragen. Zum anderen wird in processObserver() der Observer durch Aufruf der Methode check() veranlaßt, bestimmte Attributswechsel zu melden und in die KerEventList einzufügen.

Verarbeitung eines ObservationAttrEvent

Wir betrachten den Mechanismus der Übermittlung am Beispiel eines ObservationAttrEvent:
Zunächst wird vom Client (Insect oder InsectTool) aus Sim::Insect::observeAttr() mit dem Attributstyp und dem oberen und unteren Grenzwert aufgerufen. Dort wird nichts weiter getan, als im zugehörigen Observer die Operation setAttrInterval() mit dem zugehörigen Attribut als Parameter aufzurufen. Im Observer werden die Attribute mit ihren jeweiligen Grenzwerten, ober- bzw. unterhalb deren ein Event generiert werden soll, gespeichert.

Verarbeitung eines ObservationThingEvent

Der Mechanismus der Erzeugung und Übermittlung eines ObservationThingEvent ist etwas komplizierter, da seine Verarbeitung bis in die physikalische Ebene hineinreicht:
Bei jedem Durchlauf von processMove() im Kernel-Insekt wird in der zugehörigen physikalischen Referenz die Methode doObserve() aufgerufen, welche die private Eventliste (Achtung, hierbei handelt es sich um Events auf Physical-Ebene!) zurückgibt. Die Physikalische Referenz erhält ihrerseits die Events von dem Welt-Objekt PhyWorldImpl durch Aufruf von observe(). Hat das Kernel-Insekt nun die Liste erhalten, so wird diese an die Event-Liste auf Kernel-Ebene angehängt.

Übergabe der Event-Liste und Verarbeitung durch den Client

Will nun der Client die Eventliste zur weiteren Verarbeitung entgegennehmen, so muß er dies über den Aufruf der Methode `doProcess()` im `KerInsectImpl`-Objekt tun. Diese gibt die Liste zurück und erzeugt eine neue Event-Liste, die wieder die eingehenden Events bis zur nächsten Anforderung aufnimmt. Will der Client dann die empfangene Liste auswerten, so muß er die in ihr enthaltenen Events als `unions` in die in Abb. 1 gezeigten Spezialisierungen auflösen und verarbeiten. Ein Beispiel dazu findet sich in `InsectTool.java`.

3 Events auf der `Physical`-Ebene

Auf der physikalischen Ebene sind Events allein für die Darstellung von Objekten aus der Sicht des jeweiligen Thing zuständig. Sie haben dabei eine ähnliche Form, wie die `ObservationThingEvents`, in die sie auf Kernel-Ebene umgewandelt werden.

```
//idl
struct Event {
    Physical::Thing  Thing;
    long             Index;
    Sim::Position    Pos;
    Sim::Volume      Vol;
  };
```