



ARBEITSGRUPPE
SOFTWARETECHNIK
(INSTITUT FÜR INFORMATIK)



ARBEITSGRUPPE
INGENEURPSYCHOLOGIE
(INSTITUT FÜR PSYCHOLOGIE)

Das ATEO Automaten Framework

Nicolas Niestroj

Agenda

- » **Automatiken-GUI**
- » AAF Framework
 - » compute-Methode
 - » SamState
- » Hilfreiche Klassen
 - » DistanceDictionary
 - » RacingLine

Automatiken-GUI

Konfigurationstool der Automatik-Funktionen

Neu Öffnen... Speichern Speichern unter... Beenden

Elementare Funktionen

Zeige Funktionen aus Kategorie:
(Alle Funktionen)

- AAFAdaptiveSteering
- Bobbahn
- Eingabeumkehr
- Eingabeverteilung
- Halt (Debug)
- Hinweis bei Gabelungen
- Hinweis bei Hindernissen
- Mauseingriffe
- Obstacle Helper testin...
- Streckenvorschau
- VisualHintsDemo
- Visuelle Hinweise (Aydan)

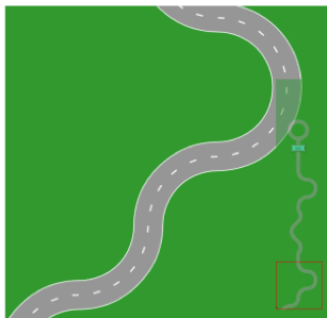
Ablaufplan

```
graph TD;
  Anfang --> Streckenvorschau;
  Anfang --> KomplexeHinweise[Komplexe Hinweise];
  Streckenvorschau --> Ende;
  KomplexeHinweise --> Ende;
```

Konfiguration von Streckenvorschau

Beschreibung

Diese Funktion blendet eine Streckenvorschau ein.



Aktivierung + ▲

Wann soll die Funktion aktiv sein?
Benutze (+) um einen gültigen Ausdruck zu konfigurieren.
Ohne Ausdruck ist die Funktion immer aktiv.

Parameter ▲

Objekte zum Anzeigen

- Zeige Hindernisse
- Zeige Objekt
- Zeige Ränder des Sichtbereichs

Größe, Position, Transparenz

Breite (px): 150 Höhe (px): 767
Abstand zum Rand von rechts (px): 1
Transparenz (%): 50

Agenda

- » Automaten-GUI
- » **AAF Framework**
 - » compute-Methode
 - » SamState
- » Hilfreiche Klassen
 - » DistanceDictionary
 - » RacingLine

AAF Framework

» Ableiten von der Klasse AAFAgent

```
AAFAgent subclass: #AAFMouseInputAgent
  instanceVariableNames: 'mwi scaleFactorX scaleFactorY'
  classVariableNames: ''
  poolDictionaries: ''
  category: 'AAF-Agents-Dev'
```

» Zu implementierende Methoden

» compute

» Wird jeden Tick ausgeführt

» plainTextName

» Name der Automatik in der Automatiken-GUI

» readyForUse

» true/false, wenn true, dann wird die Automatik in der Automatiken-GUI aufgelistet und kann verwendet werden.

» Initialize (optional)

AAF-Framework

- » Zusätzliche Methoden für die Automaten-GUI wichtig
 - » shortDescription
 - » Kurze Beschreibung der Automatik, zur Anzeige in der Automaten-GUI
 - » tags
 - » Gibt Funktionskategorie(n) an
 - » getAllProps
 - » setAllProps: aPropertyDictionary

AAF-Framework

- » konfigurierbare Automaten
 - » Variablen sinnvoll benennen (z.B. propDistance)
 - » Variable als instance-variable deklarieren
 - » getter/setter erstellen
 - » Im Konstruktor initialize die Variablen initialisieren
- » compute-Methode
 - » wird alle 39ms ausgeführt
 - » hier kommt der Quellcode für das Verhalten der Automatik hin

AAF Framework: getAllProps/setAllProps

```
getAllProps
| dict |
dict := super getAllProps.
dict at: 'mwi' put: self mwi.

^ dict
```

mwi ist ein key im Dictionary dict

Alle vom Agenten geerbten Properties
Werden geladen

Lokale Properties (mwi) werden in
Instance-Variablen gespeichert

```
setAllProps: aPropertyDictionary
```

```
(aPropertyDictionary size >= 2)
ifTrue: [
    super setAllProps: aPropertyDictionary.
    self mwi: (AAFUtils convert: (aPropertyDictionary at: 'mwi') type: 'Number').
].
```

Auf den richtigen Typ achten, evtl. Konvertierung in String

AAF Framework: GUI Integration

- » Ableiten von AAFGTDialog
 - » Konvention: Name der Automatik mit Postfix „Dialog“
- » Instanzvariablen der Properties mit gleichem oder ähnlichen Namen
- » Nötige Instanzmethoden
 - » addSpecialElements
 - » Enthält die Anzeigeelemente
 - » updateFromDelegate
 - » Sorgt für das korrekte Anzeigen der eingestellten Properties
 - » userChose: property

AAF Framework

» AAFGT Widget Gallery

» Kann aufgerufen werden mit:

AAFGTWidgetGallery open



AAF Framework

» Auflisten der Anzeigeobjekte

addSpecialElements

"Sets up parts which are unique for this type of dialog."

| *container* |

dropDownMenu := AAFGTDropDownMenu

value: 'MWB1'

items: #('MWB1' 'MWB2')

target: **self**

action: #userChose:.

container := AlignmentMorph newRow.

container

color: **Color** transparent;

addMorphBack: (AAFGTOneLineLabel contents: 'Maussteuerung aktivieren für: ');

addMorphBack: dropDownMenu.

self buttonArea

addMorph: (AAFGTWidgetUtils centered: *container*).

self updateFromDelegate.

[

AAF Framework

- » `updateFromDelegate` holt die aktuell gesetzten Einstellungen der Automatik und übergibt sie der GUI zum Anzeigen

`updateFromDelegate`

```
| value |  
super updateFromDelegate.  
  
value := delegate mwi = 1  
    ifTrue: [ 'MWB1' ]  
    ifFalse: [ 'MWB2' ].  
  
dropDownMenu label: value.]
```

AAF Framework

- » userChose: setzt die vom Benutzer eingestellte Eigenschaft der Automatik

```
userChose: mwi

mwi = 'MWB1'
  ifTrue: [ delegate mwi: 1 ].
mwi = 'MWB2'
  ifTrue: [ delegate mwi: 2 ].

self updateFromDelegate.
```

AAF Framework: SamState

- » Schnittstellenobjekt zwischen den Automaten und SAM mit folgenden Feldern:
 - » bitBlts
 - » Array mit Bildern, die gezeichnet werden sollen
 - » directSetPowerMWI1/2
 - » Einflussverteilung (50-50) der MWI 1 bzw. 2
 - » joystickRaw1/2
 - » Joystickdaten der MWI 1 bzw. 2, welche direkt vom Joystick kommen ohne Verarbeitung durch SAM
 - » visualHint
 - » Art des visuellen Hinweises, den man zeigen möchte
 - » trackingState
 - » Referenz auf das SAMModelData-Objekt
 - » centerOfTrackingObject
 - » Koordinate der Mitte des Fahrobjektes auf dem Bildschirm (nicht auf der Strecke)

Agenda

- » Automaten-GUI
- » AAF Framework
 - » compute-Methode
 - » SamState
- » **Hilfreiche Klassen**
 - » DistanceDictionary
 - » RacingLine

Hilfreiche Klassen: Distanzen

- » Informationen über die Entfernung zu bestimmten Streckenteilen
- » SAMControllerDistance
 - » distanceToNextFork: aSymbol
 - » distanceToNextCurve: aSymbol
 - » distanceToNextObstacle: aSymbol
 - » aSymbol: #nextLeftCurve30, #nextElrFork, #nextRllFork, #nextDynamicObstacle

Hilfreiche Klassen: RacingLine

- » AAFSupportRacingLine (in der Kategorie AAF-Agents-Concepts-Support)
- » Für einen Parameter Y erhält man den entsprechenden X-Wert der RacingLine
 - » racingLine: yValue
 - » racingLineReverted: yValue
 - » racingLineFork: yValue → x-Wert Array
 - » racingLine: yValue1 and: yValue2
 - » racingLine:yValue Branch: symbol
 - » Symbol: #left bzw. #right
 - » racingLine: yValue to: yValue2 → x-Wert Array für das Intervall yValue1 – yValue2
 - » racingLine:yValue1 to: yValue2 Branch: Symbol → x-Wert Array des Intervalls für den ausgewählten Zweig

Vielen Dank

» Fragen?