

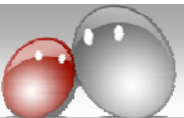
A mobile fitness Companion

Olov Ståhl, Björn Gambäck, Preben Hansen

Swedish Institute of Computer Science

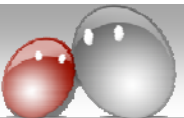
Markku Turunen, Jaakko Hakulinen

University of Tampere



Overview

- A platform for mobile Companions
- Implementation of a Health and Fitness scenario
- Uses an interface based on graphical elements as well as speech
- Is part of a larger system focused on Health and Fitness (Companions project)
- It is not an attempt to build "yet another" mobile fitness system

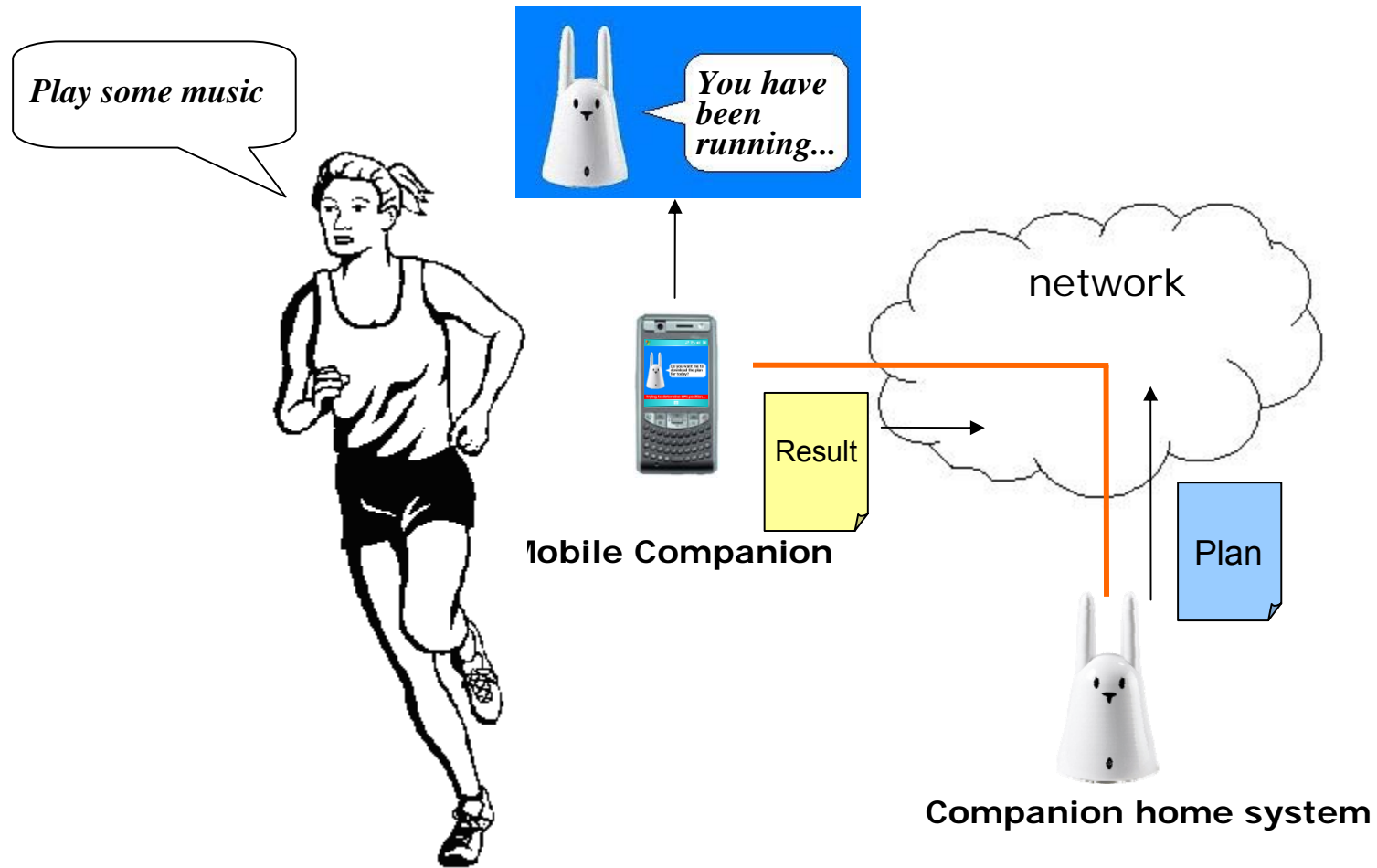


The user interface

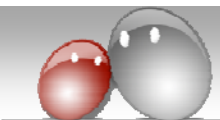
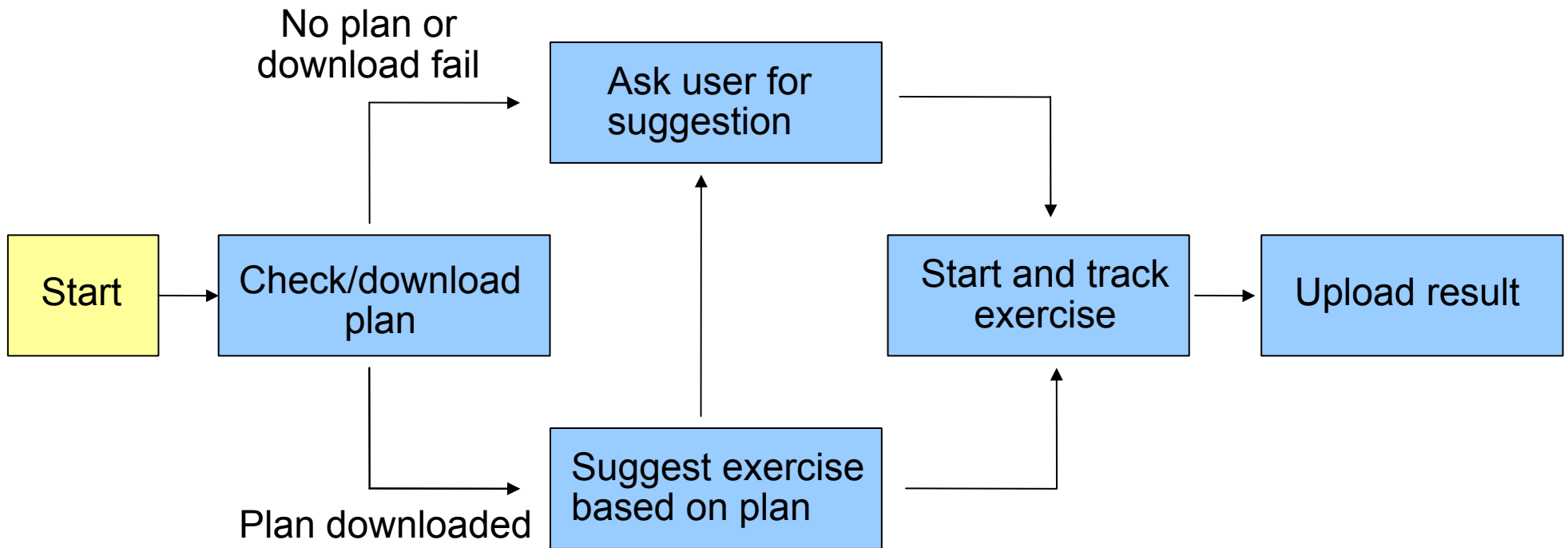
- Single screen
 - Nabaztag rabbit image
 - Speech bubble
 - Message bar
- Same voice as home system



Integration with home system



Dialogue "flow"



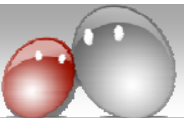
Plan document

- An XML document containing a list of activities
- Travel
 - From
 - To
 - Mode of "transportation"
- Exercise
 - Type
 - Time



System exercise suggestion

- Find out where the user is (home, work, gym, ...)
- Check the time of day
- Find the first pending exercise that matches location and time
- Suggest exercise
- If user rejects suggestion, repeat



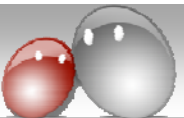
User exercise suggestion

- Simply ask the user to select an exercise from a list of available types
 - Jogging
 - Walking
 - Cycling



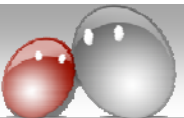
Exercise execution

- Start tracking pace, distance, time and calories burned using GPS information
- Display tracking data on the GUI
- Accept a few commands
 - Status
 - Music playback
 - Exercise complete



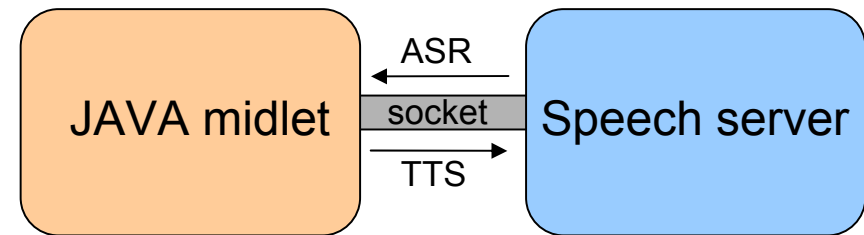
Post exercise

- Summarize exercise result
- If user agrees, create result document and upload it to home system



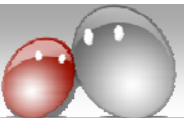
Implementation

- Windows Mobile devices
- 2 programs are running on the device:
 - a Java midlet
 - a speech server (C++)
- The midlet sends ASR/TTS commands, the server returns results
- ASR is activated via a keyboard button press



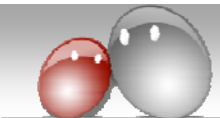
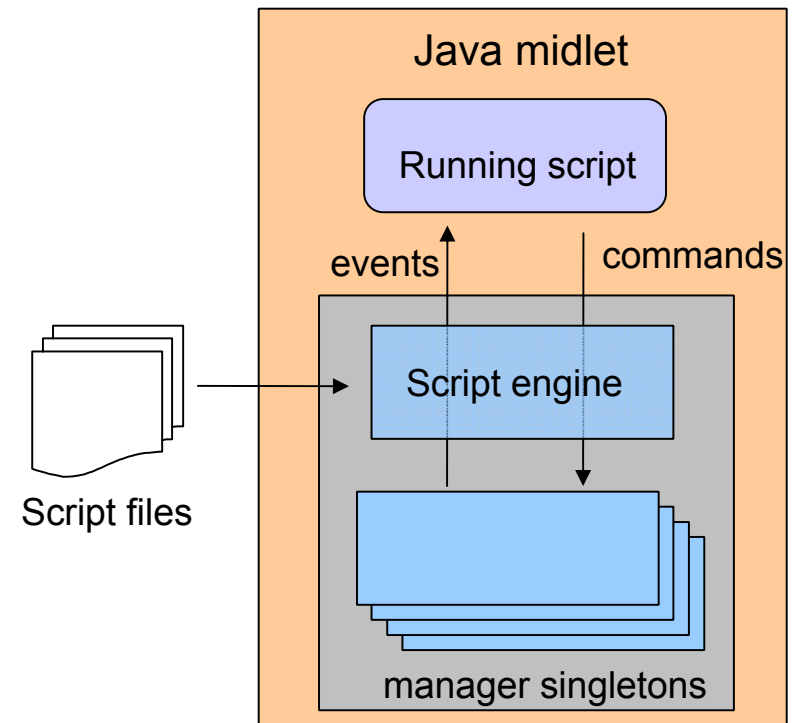
The Java midlet

- Handles the GUI and implements the application logic (dialogue control, etc)
- Input via voice (speech server ASR), buttons and screen "clicks"
- Output via voice (speech server TTS) and text
- Tracks the exercise progress (distance, pace, calories burned) by the use of a GPS
- Supports audio playback
- Can download plans (over the Internet) from the home system, and upload exercise results



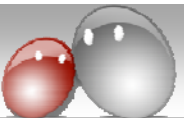
Midlet scripting engine

- Scripts implements the application logic
- Java layer provides basic "services" (e.g., GPS, ASR/TTS, data store)
- Scripts receive events from Java managers, and can invoke services via script commands
- Scripts can be read from the local file system or the web

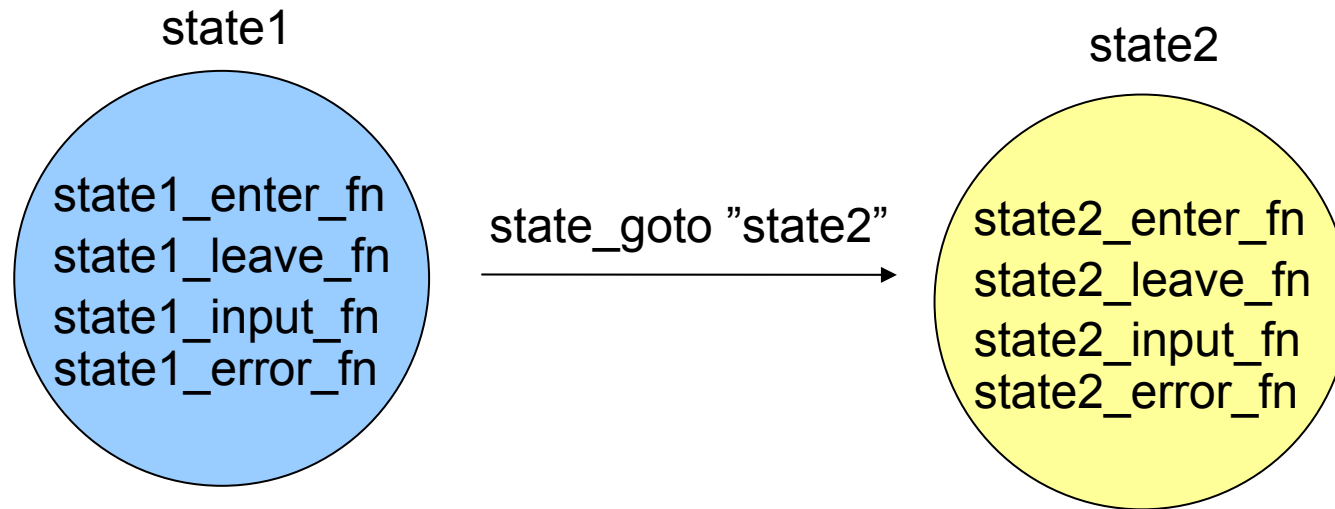


Dialogue management

- The dialogue is handled by scripts running in the scripting engine
- Each dialogue script defines a state in a state machine
 - An input method for analysing ASR input (utterance, confidence, semantic info)
 - A grammar "describing" expected user input
 - Script commands for moving between states
 - TTS output hard-coded in each state



Dialogue states



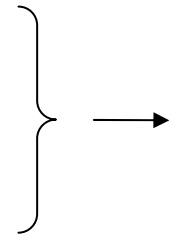
- States must implement a number of pre-defined procedures
- The state procedures are called automatically by the system
- There are specific script commands for state transitions, grammar selections, tts output, etc. They have to be called explicitly by the script programmer.
- All state procedures are "normal" script procedures and can use all features of the scripting language (variables, if-then-else, procedure calls, etc.).



Dialogue script example

```
##  
# This procedure is called when entering this state  
#  
proc state_selectActivity_enter {} {  
    asr_grammar_select "activitySelect"  
  
    if {ne [getprop "session.last.activity"] ""} {  
        state_goto "suggestActivity" "" ""  
    } else {  
        tts_out "So what do you want to do today?" ""  
    }  
}  
  
##  
# This procedure is called when leaving this state  
#  
proc state_selectActivity_leave {} {  
    puts "state_selectActivity_leave"  
}  
  
##  
#  
#  
proc state_selectActivity_reset {} {  
    puts "state_selectActivity_reset"  
}  
  
.....
```

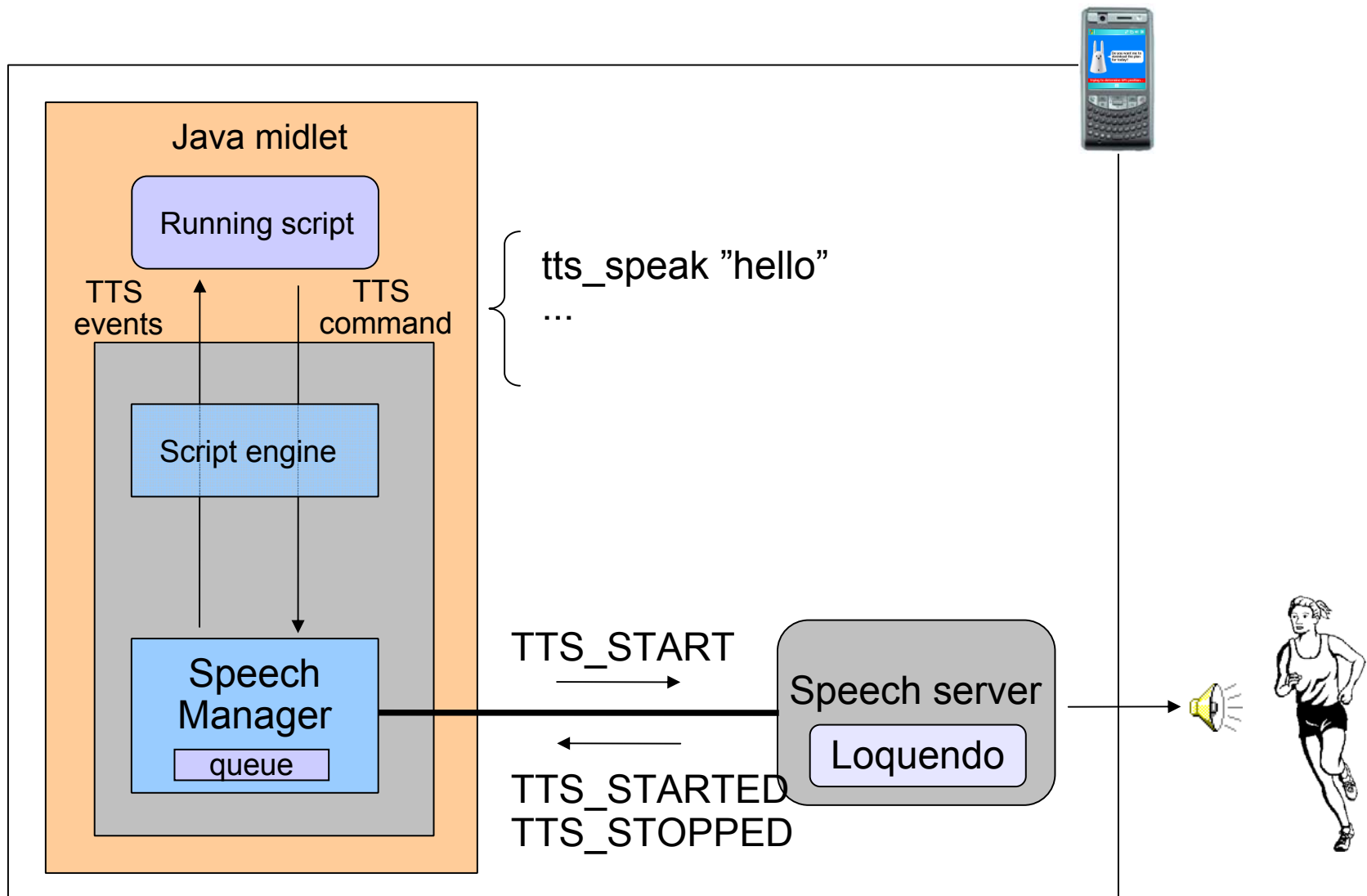
state_selectActivity.hcl



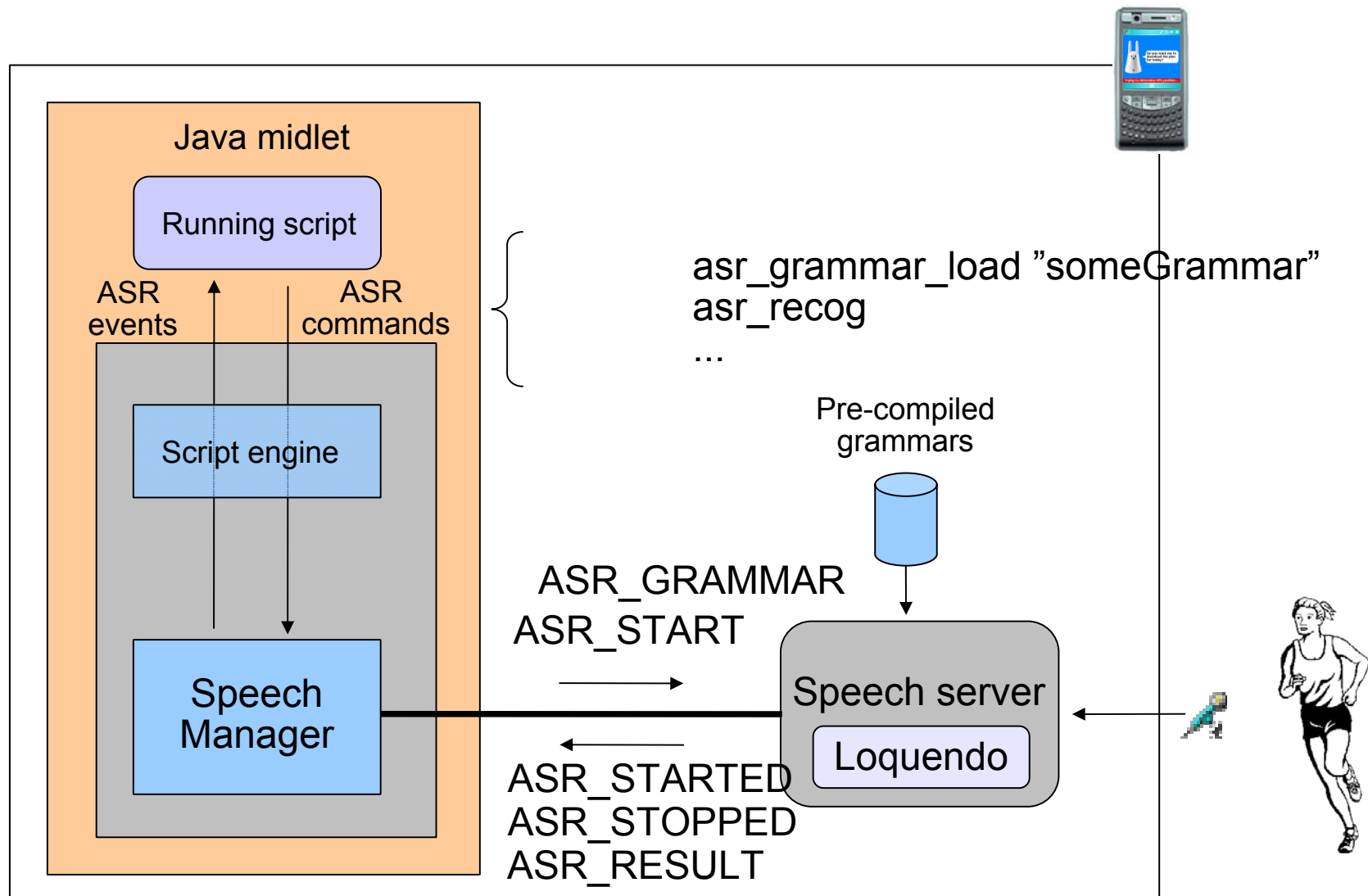
```
##  
# This procedure is called when entering this  
state  
#  
proc state_selectActivity_enter {} {  
  
    asr_grammar_select "activitySelect"  
  
    if {ne [getprop "session.last.activity"] ""} {  
        state_goto "suggestActivity" "" ""  
    } else {  
        tts_out "So what do you want to do today?" ""  
    }  
}  
}
```



TTS overview

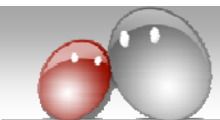
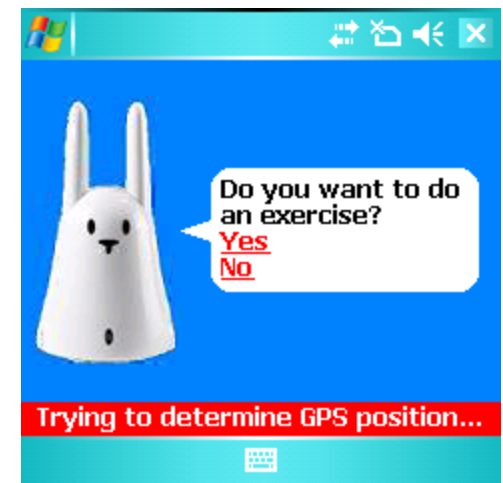


ASR overview



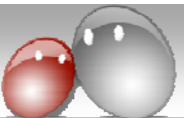
Multi-modal input

- Script programmers can map key presses and screen clicks to input passed on to input handlers
- Allows the "same" input to be provided by different means (asr, keyboard, screen)
- Suitable for "simple" input such as yes/no answers, list selections, commands, etc
- Doesn't scale



Data storage

- Data can be stored in the device's record store
- Data can also be stored/fetched on a remote server
- The store is dynamic, i.e., can be modified during runtime
- Allows for persistent data
 - User info (name, weight, age, ...)
 - Plan
 - Exercise results



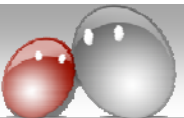
SpeechServer

- Performs ASR/TTS operations on request by the midlet (select grammar, select voice, TTS output, ASR input)
- Sends back events such as
 - TTS_START and TTS_STOP
 - ASR_START, ASR_STOP and ASR_RESULT
- Loquendo Embedded ASR 7.4 and TTS 7.6.5
- Grammars must be pre-compiled and installed on the mobile device
- Speech Recognition Grammar Specification 1.0 GRXML grammars + Semantic Interpretation for Speech Recognition 1.0 tags (**currently strings only**)

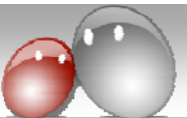


Conclusion and future work

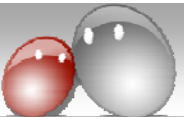
- Currently a platform for experimenting with mobile Companions
- Extend the H&F scenario, e.g., support other types of exercise activities
- Integration with other Companion demonstrators
- (Re)move hard-coded TTS messages, support multiple languages
- Sensors for "gesture" input



Questions?



Example dialogues



Exercise selection

C: Hello, do you want me to download today's plan?

U: Okay

C: Please wait while I download the plan

[plan is downloaded]

C: Plan downloaded. I found two exercise activities.

C: What's your current location?

U: I'm at home

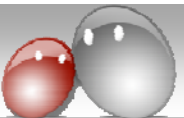
C: There is a planned activity to go cycling to work.

C: Do you want to do this exercise?

U: I'd rather walk instead

C: You would like to walk to work, correct?

U: Yes



Exercise tracking

C: Ok. Select start when you are ready to begin

U: Start [button press or voice input]

C: Exercise started

[Time passes]

U: Play some music

[Music starts]

U: Status [button press or voice input]

C: You have been walking for 12 minutes, distance is 1.1 kilometers ...

[Time passes]

U: Stop

C: Exercise stopped. You have been walking for 34 minutes ...

C: Do you want me to upload the exercise result?



Demo during lunch

