Scripts attached to answers

Scripts can
  - schedule a sequence of utterances
  - ... or VHMSG commands
  - update variables
  - modify utterances
Scheduling

- **State Machine**
  - Behavior
    - a thing that does things
    - send event, wait for some time, a collection of behaviors, etc.
    - fires events when starts and ends
  - Trigger
    - responds to events, starts behaviors
  - Event
    - external and internal
    - a structured object
  - Context
    - stores triggers

- **Machine listens and processes events**
- **The current execution state: triggers + scheduled and executing behaviors**
- **A state can be paused, saved, or restored**
Utterance Script Example

```javascript
engine.context.presentCharacters.add("rio");
behavior1 = engine.schedule("rio-whats_goin_on");
engine.schedule("utah-psst_ranger");
engine.schedule("utah-ye_r_smart_now_try_and_stay_alive");
engine.schedule("utah-no_way_rio_kin_miss");
engine.schedule("harmony-them_sweat_rings_and_dust", behavior1.done);
engine.schedule("rio-get_packed_harmony");
engine.schedule("harmony-i_dont_want_to_go_to_wichita");
engine.schedule("rio-i_like_shootin_things");
engine.schedule("rio-why_dressed_so_nice");
engine.rest(1.5);
engine.schedule("rio-wait_hold_still");
engine.rest(1.5);
engine.schedule("rio-talking_about_right_there");
behavior2 = engine.schedule("rio-collect_tin_badges");
engine.schedule("utah-hes_lying_to_you");
engine.schedule("harmony-kill_you_anyways", behavior2.done, 0);
engine.schedule("rio-theres_the_truth");
engine.rest(60);
engine.schedule("sequence-rio-rio_calls_utah");
```
Scriptable Dialogue Manager

- **Three main components**
  - State machine
  - Utterance scripts
  - Dialogue manager script

- **Model**
  - Utterances
    - you can modify them!
  - Categories, tokens, domains
  - Other variables
    - who is present
    - is the gun out?
    - where the gun is pointing at?
  - Classifiers
Main DM Script

- Configures the State Machine
  - sets event handlers, timers, model, etc...

```java
public boolean npcBehavior_done(Event event) {
    global.lastCharacterStoppedSpeaking = System.currentTimeMillis();
    if (event?.utterance?.toss)
        return handleToss(event);
    return false;
}

public boolean npcBehavior_begin(Event event) {
    global.lastCharacterStartedSpeaking = System.currentTimeMillis();
    return false;
}

public boolean vrSpeech_start(Event event) {
    global.lastUserStartedSpeaking = System.currentTimeMillis();
    return false;
}

public boolean handleEvent(Event event) {
    global.lastUserStoppedSpeaking = System.currentTimeMillis();
    return false;
}
```
Main DM Script

- Can have multiple configurations
- The State Machine can paused, reconfigured, and resumed
- In Groovy, the configurations defined as classes, so we can take advantage of inheritance
  - a set of handlers can shared by using a common base class
- Example. In Gunslinger
  - there are 3 configurations
    - normal conversation
    - gun out
    - shoot out
  - the last two do not handle user speech input
Three notions of state
- Dialogue state (state)
  - a set of rules
  - how the answers are selected
  - regular conversation vs. gun out vs. gunfight
- Classifier state (domain)
  - what answers are selected
  - how to map user’s text onto character text
- Output state (behavior)
  - how answers are outputed
API (Engine.java)

- List<Map<String,Object>> search(String inVHName, Map<String,Object> inQuery);
- List<Map<String,Object>> filter(List<Map<String,Object>> inList, Condition<Map<String, Object>> inFilter);
- void postEvent(Event inEvent);
- Behavior rest(double inDelayInSeconds);
- Behavior rest(double inDelayInSeconds, Event inEvent);
- Behavior schedule(String inUtteranceID);
- Behavior schedule(String inUtteranceID, Event inEvent);
- Behavior schedule(Map<String, Object> inUtterance);
- Behavior schedule(Map<String, Object> inUtterance, Event inEvent);
- Behavior schedule(Runnable inBehavior);
- Behavior schedule(Runnable inBehavior, Event inEvent);
- Behavior scheduleInResponse(Map<String, Object> inUtterance, Event inSourceEvent);
- Behavior send(String inCommand);
- Behavior send(String inCommand, Event inEvent);
- Executor getExecutor();
- Trigger addTrigger(Trigger inTrigger);
- Trigger addTrigger(Event inEvent, Runnable inRunnable);
- boolean removeTrigger(Trigger inTrigger);
- void reset();
- Collection<Map<String, Object>> answers();
- Collection<Map<String, Object>> answersForCharacter(String inName);
- int seenRecently(Map<String, Object> inUtterance, int inHistoryLength);
- boolean message(String inCommand);
- void begin();
- void end();
public boolean vrSpeech_asr_complete(Event event) {
    if (!event.text || ((String)event.text).trim().length() == 0) return false;

    if (global.domainName == null)
        global.domainName = (String)event.speaker;

    try {
        List<Map<String,Object>> answers = engine.search(global.domainName, event);

        if (answers.isEmpty()) {
            sendOfftopic();
        } else {
            Map<String,Object> selectedUtterance = (Map)answers[0];

            if (selectedUtterance.type == Global.kAlternative) {
                if (!global.lastSpeechEvent) {
                    sendOfftopic();
                } else {
                    offTopicCount = 0;
                    return vrSpeech_asr_complete(global.lastSpeechEvent);
                }
                return false;
            }

            if (selectedUtterance.type == Global.kRepeat) {
                if (!global.lastUtterance) {
                    sendOfftopic();
                } else {
                    sendRegular(global.lastUtterance);
                }
                return false;
            }

            global.lastSpeechEvent = event;
            sendRegular(leastRecent(answers, Global.kHistoryWindow));
        }
    } catch (Throwable t) {
        t.printStackTrace();
    }

    return false;
}
Multiple state machines:
  – main
  – one for each utterance sequence

Events go to all machines in parallel

SCXML
  – and JavaScript

The result: states and executable code are separate