AGTk-based Shared Movie Player

Robert Olson

The Problem

- Synchronized viewing of a movie file at multiple sites

Challenges

- Determination and definition of “session”
- Movie file distribution
- Synchronization of start/stop/change position

Sessions

- A single instance of a shared movie playing session
- State:
  - Current movie being played
  - Current position in the movie
  - Current state (playing, paused, stopped, etc)

Sessions, cont.

- Session represented as an AGTk application object
  - App obj holds a dictionary containing state, and an event channel for realtime communication

Session data

- currentMovieType
  - Currently, only supports movies in the datastore
- currentMovieStore
  - URL to the datastore holding the movie
- currentMovieFile
  - Filename of the movie in the datastore
- state
  - Current state (playing, stopped, etc)
- position
  - Current position in the movie (in seconds)

File distribution

- Movie files placed in Venue datastore
- Session state used to retrieve current file
- Clients’ loading new file causes event to be distributed notifying clients of new file
- Clients load the files using the API in the DataStoreClient module.

State Synchronization

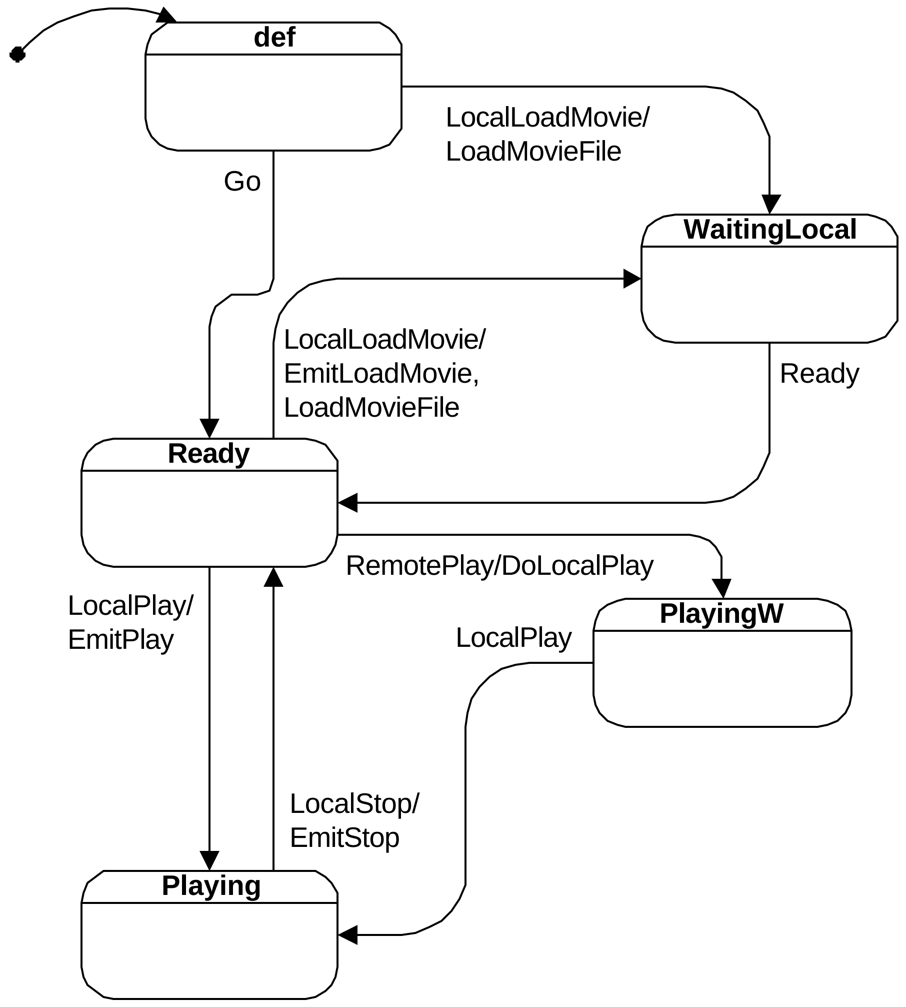
- Venue event channel used to distribute state update messages:
  - status_changed
    - Emitted when a user starts or stops the movie
  - new_movie
    - Emitted when a user loads a new movie file
  - position_changed
    - Emitted when a user changes the position of a movie playback

Client Considerations

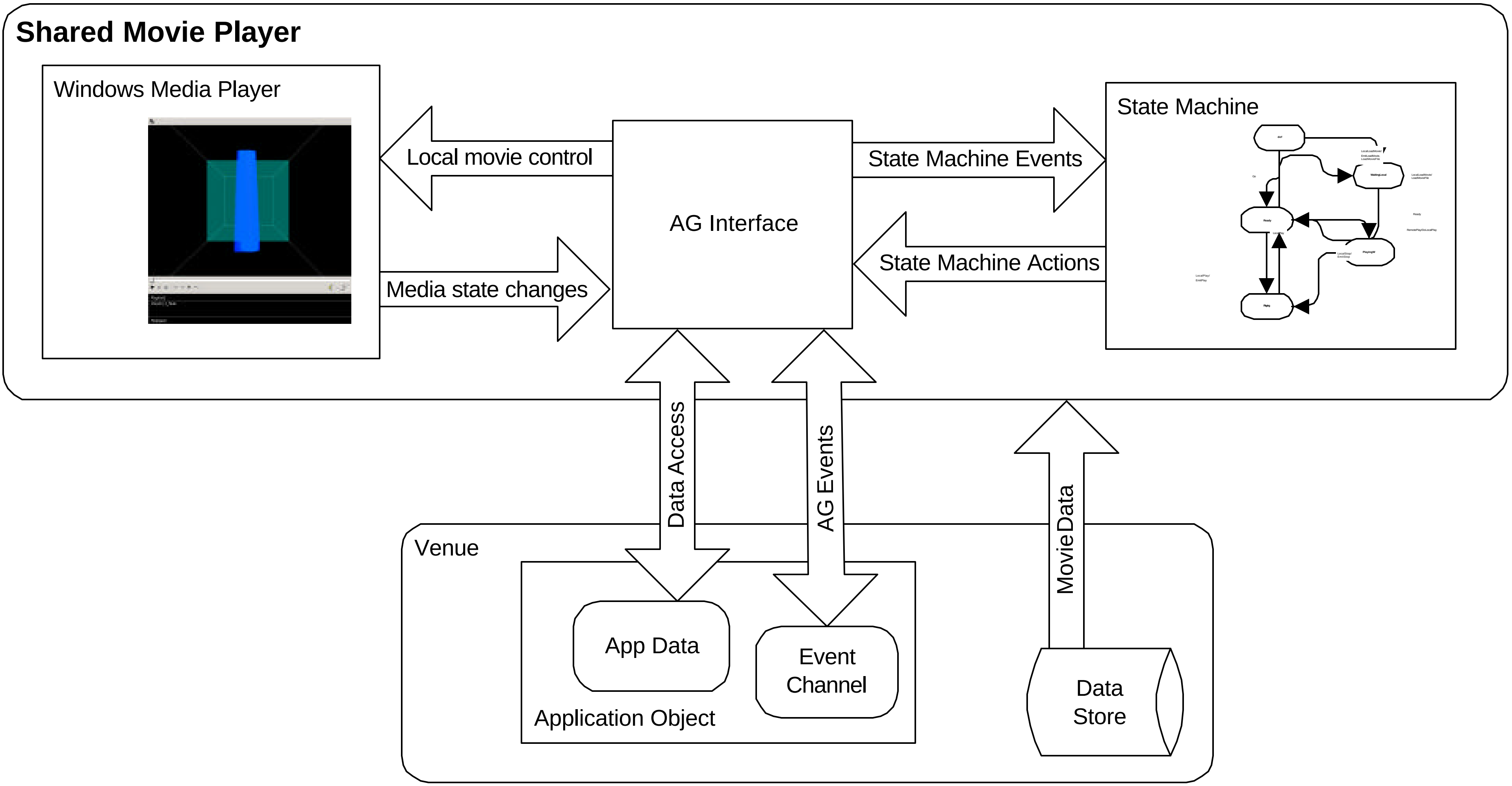
- On Windows, video rendered by embedded Windows Media Player
- Uses native media player controls (play, stop, set position)
- Complication: programmatically generated media player control generates local feedback that’s the same as manual control

Client, cont.

- Complexities of client state manipulation handled with an explicit state machine



AG Integration



Distributed State

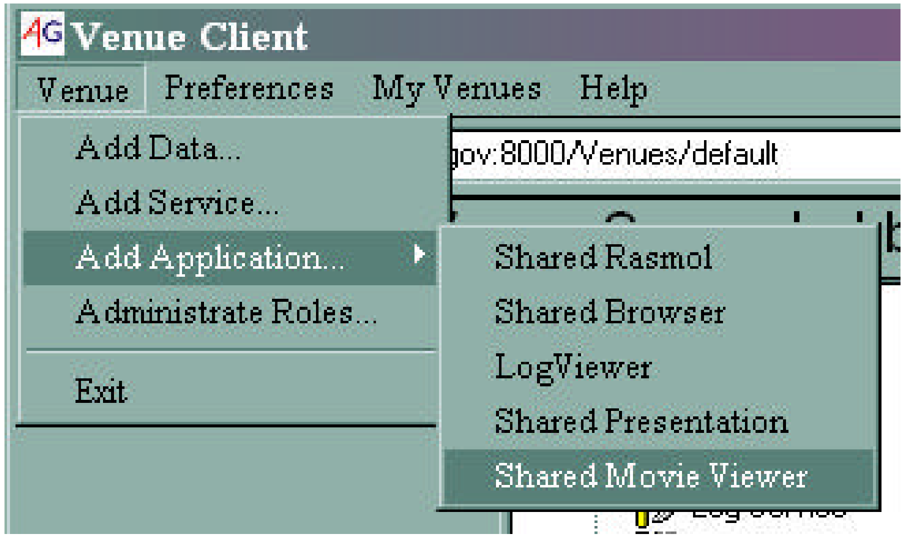
- Event channel used to distribute effects of user changes (Play, Stop, Load, etc)
- Potential for races, etc
  - No global ordering of messages
- What effect does this have?
  - Possible confusion if multiple people manipulate state at the same time
- Social protocols should help
- Full distributed control algorithms would help more, but require strong ordering semantics in communications

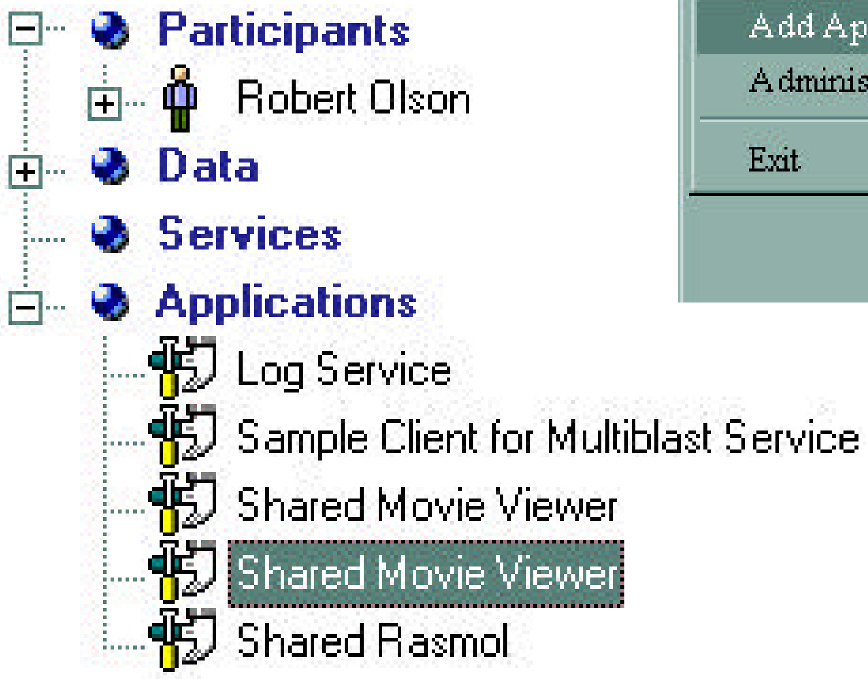
Membership

- Calculation of membership is approximate (see previous slide on event channel semantics)
- Used as advisory means: user feedback
- Mechanism: new events:
  - client_join (on first joining)
  - client_present (keepalive)
  - client_leave (on exiting)

App startup

- Application object is created in the venue:





App startup, cont.

- Application developer writes application description file:

```
[application]
name = Shared Movie Viewer
mimetype = application/x-ag-movie-viewer
extension = agmovie
files = sharedmovie.bat
[commands]
Open = sharedmovie.bat %(venueUrl)s %(appUrl)s
```

- Defines the “Open” command on the shared movie item that…
- Invokes sharedmovie.bat with the venue’s SOAP URL and the app object’s SOAP URL

App startup, cont.

- My apps use a batchfile for startup:

```
cd C:\Program Files\Access Grid Toolkit Applications\Shared Movie
   Viewer
C:\Python22\\python.exe smc.py %1 %2 %3 %4 %5 %6 %7 %8 %9
```

- Registered with agpm.py:

```
agpm.py –f sharedmovie.app
```

- Wrapped up with an InnoSetup installer
- But I might be odd ☺
- Can distribute zipfiles of code which are installed directly via ‘agpm.py –z file.zip’

App startup, cont.

- App extracts URLs and retrieves information from venue and app object:

```
venueURL = sys.argv[1]
appURL = sys.argv[2]
datastore =
  DataStoreClient.GetVenueDataStore(venueURL)
app = Client.Handle(appURL).GetProxy()
publicId, privateId = app.Join()
```

What’s next?

- Streaming video support ?
- Tighter synchronization (avoid net-lag problems)
- Synchronized playback with Tiled Display movie players
- Linux support (mplayer? xanim?)