Montana Tech

Science Mine

Digital Signage Project

Finite State Machine Description

Author:
Phillip J. Curtiss

Principal Investigator:
Fred Hartline

January 18, 2017

1 Introduction

The Science Mine Digital Signage project provides a self-guided signage system for museum exhibits whereby visitors may interact with a variety of input devices to allow for the exploration of the content within the signage system, in a similar way that a visitor to the exhibit is encouraged to learn by exploring through interacting with the exhibit. The variety of different input devices allows visitors of different ages and different knowledge levels to interact with a single signage platform that adapts to provide information that is relevant to what is of interest to the visitor.

To accomplish this task, the signage system is comprised of the following components:

Input Devices: The current set of input devices consists of a bluetooth trackpad and a bluetooth array of buttons driven by an Arduino-based microcontroller. In the future, an additional touch screen is likely to be added. In general, however, any input device that is bluetooth HID compliant should be able to be integrated into the system without much effort.

Compute Platform: The compute platform selected is an extremely small form-factor full-function Linux-based computer made by Intel known as the Stick. In particular, we are using model STK2MV64CC which is quite a capable compute device in a small form-factor that can be connected to a high-definition projector and out of the way of visitors to the exhibit.

Projector/Display: The initial deployment will be a projector to provide a large enough display surface to show high-quality video content, and also provide high-quality thumbnails of next-level video content that may be selected by the visitor. In the future, additional touchscreen monitors might be used, or added, to provide a different presentation format and means of interaction.

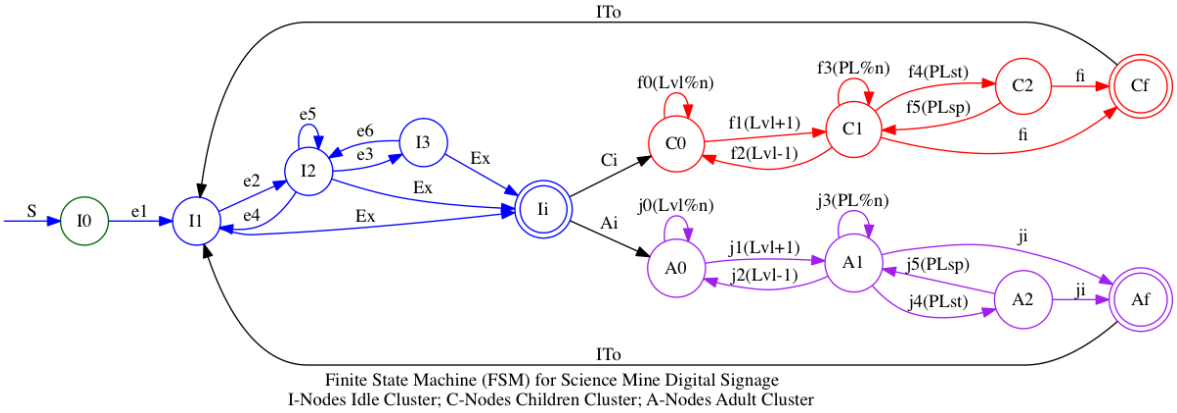
Video Content: The video content is high-quality custom produced content highlighting the use of the exhibit associated with the signage. The video content collection is organized in a graph structure that matches a given organizational structure (learning paths) through the video content collection. In this way, a single video may be included in different organizational unit structures represented within the graph - i.e. be part of multiple learning paths through the collection.

Signage Software: Custom written software that consists of the following components:

- **Single Page Web Application:** Based on the Ember application framework, the single page web application is resident in the client browser running on the Intel Compute Stick. The web application is responsible for implementing the Finite State Machine (FSM) described within this document by mapping different bluetooth HID input (from a variety of devices) to transitions within the FSM. The web application will also enter a default *idle behavior* when the is no interaction from a visitor.
- **Exhibit Object Model (EOM):** The Exhibit Object Model (EOM) is a JSON formatted description of the signage content for a given specific exhibit. This EOM is used by the single web page application to *drive* the signage. The EOM contains references to the entire video content collection and provides the organizational structure of this content. In addition, the meta-information for each video within the collection is stored within the EOM to implement textual overlays to provide descriptive text associated with the video, attribution for the video, and the like.
- **HTML:** The web application is a simple web page written using HTML5. The base-page merely has placeholders for the main background video content, a menu of all the video content for the given level, a region that overlays the background video, and zero or more regions in which thumbnail video content will play, each one representing a topic at the current level that may be explored.
- **CSS:** The cascading style sheet (CSS) is used to provide layout functionality and look and feel behaviors for the html content elements created by the web application.
- **JS:** JavaScript content is used to script and provide functional behavior of the web application.

All of these components remain the same an unchanged no matter for which exhibit they are providing signage, with the exception of the EOM. There is a one-to-one relationship between a given exhibit that has been paired with a digital signage system, and its corresponding EOM that fully describes the signage with be used with the exhibit.

Figure 1: Science Mine Digital Signage Finite State Machine



Finite State Machine (FSM) for Science Mine Digital Signage
I-Nodes Idle Cluster; C-Nodes Children Cluster; A-Nodes Adult Cluster

2 Finite State Machine

The finite state machine (FSM) for the digital signage system consists of three (3) subgraphs, I, C, A, where I describes the states and transitions associated with the system being in an idle state, C describes the states and transitions associated with the system being in a mode of providing signage-content for children, and A describes the states and transitions associated with the system being in a mode of providing signage-content for adults. All of these subgraphs are related into a single FSM in Figure 1 on page 2.

More formally:

Let I be the graph such that, $I =< I_s, I_v >$ $\quad |I_s = \{I_n \text{ for } n \in [0..3]\} \cup \{I_i\}$
$\quad |I_v = \{e_m \text{ for } m \in [1..6]\} \cup \{E_x\}$

Let C be the graph such that, $C =< C_s, C_v >$ $|C_s = \{C_n \text{ for } n \in [0..2]\} \cup \{C_f\}$
$\quad |C_v = \{f_m \text{ for } m \in [0..5]\} \cup \{f_i\}$

Let A by the graph such that, $A =< A_s, A_v >$ $|A_s = \{A_n \text{ for } n \in [0..2]\} \cup \{A_f\}$
$\quad |A_V = \{j_m \text{ for } m \in [0..5]\} \cup \{j_i\}$

Let I_0 be the start state

Let C_0, A_0 be the respective start states for the subgraphs defined by C and A respectively.

2.1 System Startup Behavior $\delta(I_0, e_1) \mapsto I_1$

Startup behavior should perform the following operations:

- Ready any application frameworks
- Ready any CSS
- Ready any JavaScript
- Initialize the Document Object Model (DOM)
- Read the EOM and prepare associated data structures
- Enter fullscreen mode (via scripting)
- Register a callback for any keyboard (HID) event from any DOM element
- Enter Idle Mode

2.2 Idle Mode Startup Behavior

Idle mode behavior is dictated by whatever state and input is provided to that state. Below is a description of each combination of state from the Idle subgraph, and associated input/event:

$\delta(I_1, e_2) \mapsto I_2$**:** Once in I_1 we know the application state has been initialized and all EOM and DOM objects are ready for transformation. In I_1 we perform the following:

- Select a Playlist P_k from the Top Level (Category); Selection model should be *shuffle* with a least-used-first algorithm.
- Ready any DOM elements in preparation for playing an element from the selected playlist
- For each video $v_i \in P_L$, place a video container D_i over (on top of) the background container of the browser window and load v_i in D_i and begin playing the video.
- Plaint the boarder of every D_i either red or white depending on the corresponding property in the EOM for v_i loaded in D_i as to whether it is content for a *child*, which should be red, or content for an *adult*, which should be white.
- Set the global mode of the *selection mode* to be *child*.
- Set a timeout counter T_i that will be used when in state I_2 to return to this state and select a different Top Level playlist

$\delta(I_2, e_4) \mapsto I_1$**:** The timeout T_i has been reached in state I_2 causing a return to I_1 where the operations are performed from the above section.

$\delta(I_2, e_5) \mapsto I_2$**:** Shuffle through the videos in the selected playlist and select a video to play v_i in state I_3 using a least-used-first algorithm. If the timeout T_i is reached or exceeded in this state, return to state I_1 for the selection of a different playlist shuffled. Decorate the boarder of D_i in which the selected v_i is contained to be a pulsating thin to thick version of its boarder color.

$\delta(I_2, e_3) \mapsto I_3$**:** Once a video v_i has been selected from the selected playlist, dwell for a programmable amount of time. When this time is reach, set the opacity of all D_i to zero in such a way to implement a programmable fade time, and simultaneously load v_i into the background container of the browser window and begin planning the video.

$\delta(I_3, e_6) \mapsto I_2$**:** Once the selected video v_i has finished playing in the background container, modify opacity of all D_i from zero to some programmable amount to make them visible again. Return to state I_2.

$\delta(\{I_1, I_2, I_3\}, E_x) \mapsto I_i$**:** Event E_x is any event resulting in any detectable HID input by the application. Once detected, transition from the shown states to the final state for the idle subgraph I_i.

Once in I_i, perform the following operations:

- Set the opacity of all D_i to zero in such a way to implement a programmable fade time
- Empty the video in the background container and replace with an image defined in EOM
- Unregister callback for any keyboard (HID) event from any DOM element
- Populate a carousel menu with all possible videos from EOM, with each D_i decorated as to whether it is content for a *child*, which should be red, or content for an *adult*, which should be white
- Hide the menu container
- Register a callback for any mouse event that appears within a certain number of pixels away from the top boarder of the window as specified in the EOM - which will then show the menu.

- Set a global idle timeout IT_0 that will be used to return to idle subgraph is no interaction is detected within the timeout period
- if the *selection mode* has been set to child, transition to $\delta(I_i, C_i) \mapsto C_0$
- if the *selection mode* has been set to adult, transition to $\delta(I_i, A_i) \mapsto A_0$

2.3 Exploring Signage Content for Children

The children content mode of the digital signage solution is designed to allow children visitors to explore the signage content using simplified user input devices. The following sections describe the states the system can be in and the associated behavior.

$\delta(C_0, f_0) \mapsto C_0$: Once in C_0 we know the application state has been initialized and all EOM and DOM objects are ready for transformation. In C_0 we perform the following operations:

- Select a Playlist P_k from the Top Level (Category); Selection model should be *shuffle* with a least-used-first algorithm.
- Ready any DOM elements in preparation for playing an element from the selected playlist
- For each video $v_i \in P_L$, place a video container D_i over (on top of) the background container of the browser window and load v_i in D_i and begin playing the video.
- Paint the boarder of every D_i either red or white depending on the corresponding property in the EOM for v_i loaded in D_i as to whether it is content for a *child*, which should be red, or content for an *adult*, which should be white.
- Register callback for keyboard (HID) input over the main window and process input as follows:
  - $d(C_0, K_b) \mapsto C_0$: change the selected category forward associated with the current selection mode,
  - $d(C_0, K_g) \mapsto C_0$: change the selected category backward associated with the current selection mode,
  - $d(C_0, K_r) \mapsto C_1$:
    - if in child selection mode, selects the level (and the associated playlist P_l and transitions to C_1
    - if in adult selection mode, switches to child selection mode, and change the selected category forward associated with the child selection mode

$\delta(C_1, f_2) \mapsto C_0$: Perform the following options:

- For all the D_i change their opacity to zero in such as way as to implement a fade effect - the amount of fade specified in the EOM
- Unregister callback for keyboard (HID) input over the main window
- Transition back to C_0

$\delta(C_1, f_3) \mapsto C_1$: Once in C_1, play the video associated with the category select, and present the visitor with the next level of video elements once the category video is completed. Perform the following operations:

- for each video v_i in the playlist associated with the category selection, place a video container D_i over (on top of) the background container of the browser window and load v_i into D_i and begin playing the video.
- add an additional D_{n+1} where n is the number of videos v_i in this level. In D_{n+1}, create a thumb nail that represents the previous level

- Paint the boarder of every D_i either red or white depending on the corresponding property in the EOM for v_i loaded in D_i as to whether it is content for a child or adult.
- Register callback for keyboard (HID) input over the main window and process input as follows:
  - $d(C_0, K_b) \mapsto C_0$: change the selected category forward associated with the current selection mode,
  - $d(C_0, K_g) \mapsto C_0$: change the selected category backward associated with the current selection mode,
  - $d(C_0, K_r) \mapsto C_1$:
    - if in child select mode, and D_{n+1} is selected, $d(C_1, f2) \mapsto C_0$
    - if in child selection mode, selects the level (and the associated playlist P_l and transitions to C_1
    - if in adult selection mode, switches to child selection mode, and change the selected category forward associated with the child selection mode

$C_1, f_4) \mapsto C_2$**:** Once a video v_i has been selected from the selected playlist, dwell for a programmable amount of time. When this time is reach, set the opacity of all D_i to zero in such a way to implement a programmable fade time, and simultaneously load v_i into the background container of the browser window and begin planning the video.

$\delta(C_2, f_5) \mapsto C_1$**:** Once the selected video v_i has finished playing in the background container, modify opacity of all D_i from zero to some programmable amount to make them visible again. Return to state I_2.

$\delta(\{C_1, C_2\}, f_i) \mapsto C_f$**:** An idle timeout has occurred and we return to idle mode by transitioning to state I_1

- Set the opacity of all D_i to zero in such a way to implement a programmable fade time
- Empty the video in the background container and replace with an image defined in EOM
- Unregister callback for any keyboard (HID) event from any DOM element
- Unpopulate a carousel menu with all possible videos from EOM
- Hide the menu container
- Unegister a callback for any mouse event

2.4 Exploring Signage Content for Adults

The adult content mode of the digital signage solution is designed to allow adult visitors to explore the signage content using more sophisticated user input devices, but will also accept input from the more simplified devices. The following sections describe the states the system can be in and the associated behavior.

$\delta(A_0, j_0) \mapsto A_0$**:** Once in A_0 we know the application state has been initialized and all EOM and DOM objects are ready for transformation. In A_0 we perform the following operations:

- Select a Playlist P_k from the Top Level (Category); Selection model should be *shuffle* with a least-used-first algorithm.
- Ready any DOM elements in preparation for playing an element from the selected playlist
- For each video $v_i \in P_L$, place a video container D_i over (on top of) the background container of the browser window and load v_i in D_i and begin playing the video.

- Paint the boarder of every D_i either red or white depending on the corresponding property in the EOM for v_i loaded in D_i as to whether it is content for a *child*, which should be red, or content for an *adult*, which should be white.
- Register callback for keyboard (HID) input over the main window and process input as follows:
  - $d(A_0, K_b) \mapsto A_0$: change the selected category forward associated with the current selection mode,
  - $d(A_0, K_g) \mapsto A_0$: change the selected category backward associated with the current selection mode,
  - $d(A_0, K_w) \mapsto A_1$:
    - if in adult selection mode, selects the level (and the associated playlist P_l and transitions to A_1
    - if in child selection mode, switches to child selection mode, and change the selected category forward associated with the child selection mode

$\delta(A_1, j_2) \mapsto A_0$**:** Perform the following options:

- For all the D_i change their opacity to zero in such as way as to implement a fade effect - the amount of fade specified in the EOM
- Unregister callback for keyboard (HID) input over the main window
- Transition back to A_0

$\delta(A_1, j_3) \mapsto A_1$**:** Once in A_1, play the video associated with the category select, and present the visitor with the next level of video elements once the category video is completed. Perform the following operations:

- for each video v_i in the playlist associated with the category selection, place a video container D_i over (on top of) the background container of the browser window and load v_i into D_i and begin playing the video.
- add an additional D_{n+1} where n is the number of videos v_i in this level. In D_{n+1}, create a thumb nail that represents the previous level
- Paint the boarder of every D_i either red or white depending on the corresponding property in the EOM for v_i loaded in D_i as to whether it is content for a child or adult.
- Register callback for keyboard (HID) input over the main window and process input as follows:
  - $d(A_0, K_b) \mapsto A_0$: change the selected category forward associated with the current selection mode,
  - $d(A_0, K_g) \mapsto A_0$: change the selected category backward associated with the current selection mode,
  - $d(A_0, K_w) \mapsto A_1$:
    - if in child select mode, and D_{n+1} is selected, $d(A_1, j2) \mapsto A_0$
    - if in adult selection mode, selects the level (and the associated playlist P_l and transitions to A_1
    - if in child selection mode, switches to child selection mode, and change the selected category forward associated with the child selection mode

$A_1, jf_4) \mapsto A_2$**:** Once a video v_i has been selected from the selected playlist, dwell for a programmable amount of time. When this time is reach, set the opacity of all D_i to zero in such a way to implement a programmable fade time, and simultaneously load v_i into the background container of the browser window and begin planning the video.

$\delta(A_2, j_5) \mapsto A_1$**:** Once the selected video v_i has finished playing in the background container, modify opacity of all D_i from zero to some programmable amount to make them visible again. Return to state I_2.

$\delta(\{A_1, A_2\}, j_i) \mapsto A_f$**:** An idle timeout has occurred and we return to idle mode by transitioning to state I_1

- Set the opacity of all D_i to zero in such a way to implement a programmable fade time
- Empty the video in the background container and replace with an image defined in EOM
- Unregister callback for any keyboard (HID) event from any DOM element
- Unpopulate a carousel menu with all possible videos from EOM
- Hide the menu container
- Unegister a callback for any mouse event

3 Missing States for the FSM

1. subgraph for the menu interactions
2. switching between adult and child modes
3. subgraph for the Bluetooth button interactions
4. subgraph for the EOM administration utility