Edith

Animation System
Intermediate Report

October 7, 2013
Revised: October 7, 2013

Eric Lund
Kramer Canfield
Zeke Rosenberg
Calder Whiteley
Jon Youmans

1 *Class Design and Interaction*

One of the components of the "Edith" system, a 2D web-based computer science teaching tool, is the Animation System. The Animation System module is important because it includes the core animation frameworks and specifications of how the animations and audio will be produced. In a visually-based programming environment, this module is clearly important because it provides animations without requiring the user to have any knowledge of a scripting language or computer graphics concepts. There are two use cases for this module: creating new media and navigating existing media. The media is specified by a different sub-system which delivers the instructions to the Animation System module. The Animation System processes the instructions (assuming the instructions are delivered correctly) and delivers the animations to another sub-system to be displayed on the screen as desired.

2 *Design Specification*

2.1 Purpose

The Animation Module allows the user to render animations without knowledge of a scripting language. This is possible by applying pre-defined animation sets to image sprites. The module also allows the user to include audio with their animations. The module is intended to interpret animation instructions and render an animation sequence.

3 *Functional Requirements*

3.1 "Use Case 1: Create Media"

1. Actor
   (a) "Taker": The Edith sub-system that takes and displays our media.
   (b) "Programmer": The individual who is using Edith through his/her web browser to learn how to program.

3.2 "Use Case 2: Navigate Existing Media"

(a) Actor

i. "Drawer": The Edith sub-system providing us with sprites to animate and a list of media creation instructions.

ii. "Programmer": The individual who is using Edith through his/her web browser to learn how to program.

3.3 *Animation System Structure*

2. 'translator' or 'reader'

(a) The Animation System will be receiving instructions for animations in JSON format. This subsystem will take the instructions and actually call the functions with correct inputs. Our required interface is JSON, but more specifically a JSON entry with the following elements:

(b) {"function name" : "jump(x1, x2, y1, y2)", "Image Name": "example.png", "sound name": "soundFile.mp3"}

(c) There will be some instructions that might not have some of these fields; 'null' will be acceptable.

3. 'animator'

(a) Once we have parsed JSON inputs into instructions, the next subsystem will carry out those instructions and make calls to the HTML5 canvas. This subsystem will have several important interactions with other pieces.

(b) This system will take the raw data about instructions- with files waiting to be loaded until the canvas is ready to actually animate. For example, this system takes the instruction to move to the right, and 'remembers' the last position of an object, and the new position with x=x+5 (example). This information will be stored in a 'frame' array so that fastforward/rewind will allow the user to 'step' through the program if they desire.

(c) This system will also take the instructions and call them on the canvas. Therefore, this is where the majority of functions that will be written, and can be called if given the correct input.

4. 'Frame Array'

   (a) The Frame Array subsystem simply holds a list of frames, each frame being one step of the animation process. These will be stored in an array so that input from buttons (done by another team) will allow the user to select a time to view from, and the frame array can then grab the necessary frames and display them.

5. Interactions

   (a) The animation system will have no direct interaction with the user; it is a part of the abstraction process we are doing on the animation to make it easier to do simple animations. Therefore, we have no mock-up of our end product, as it is reliant on the canvas created for us as well as buttons and other UI aspects we are not dealing with.

   (b) The Animation System interacts with the Story Creator module, who feeds us the JSON instructions to be parsed and carried out. It also interacts with the visual editor team, which will be providing the images, sounds, etc. to be used following the JSON instructions. Finally, we will be giving our output to the (some other team here) to be painted on the canvas.

   (c) The only other requirement for the Animation system is that we are using a 3rd party library, 'OCanvas', to make animations easier (and cooler!). This library works directly with an HTML5 canvas, so there are minimal if any changes from other teams. This can be found at http://ocanvas.org/docs