Edith

Animation System
Intermediate Report

October 7, 2013
Revised: October 18, 2013

Eric Lund
Kramer Canfield
Zeke Rosenberg
Calder Whiteley
Jon Youmans

1 *Animation System Structure*

1.1 Module Components

1. Parser
   (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", "Audio File Name": "soundFile.mp3"}
   (c) There will be certain instructions that might not have specific pieces of information, such as an animation without an associated sound. In a case where this occurs, input "null" in the appropriate location.
2. Controller
   (a) Once we have parsed JSON inputs into separate instructions, the Controller subsystem will carry out those instructions and send media creation and display instructions to the HTML5 canvas. This component will have several important interactions.
   (b) The Controller will take the raw data about the media instructions and create a list of instructions to send to the HTML5 canvas for drawing and playing sounds. The instructions for the canvas will be stored in a Scene Array, which will later be useful for play(), pause(), and seek().
3. Scene Array
   (a) The Scene Array sub-component of the Controller simply holds a list of frames, each frame being one step of the animation process. These will be stored in an array so that playback control functions (called by another team) will allow the user to select a time to view from, and the Scene Array can then get the necessary media and display it.

1.2 Design Rationale

The Parser is its own sub-system rather than being part of the Controller because interpreting the input is a one-way process and the Controller does not need to access the raw inputs. The Scene Array component is a sub-component of the Controller to allow for easier access to the media instructions in the array.

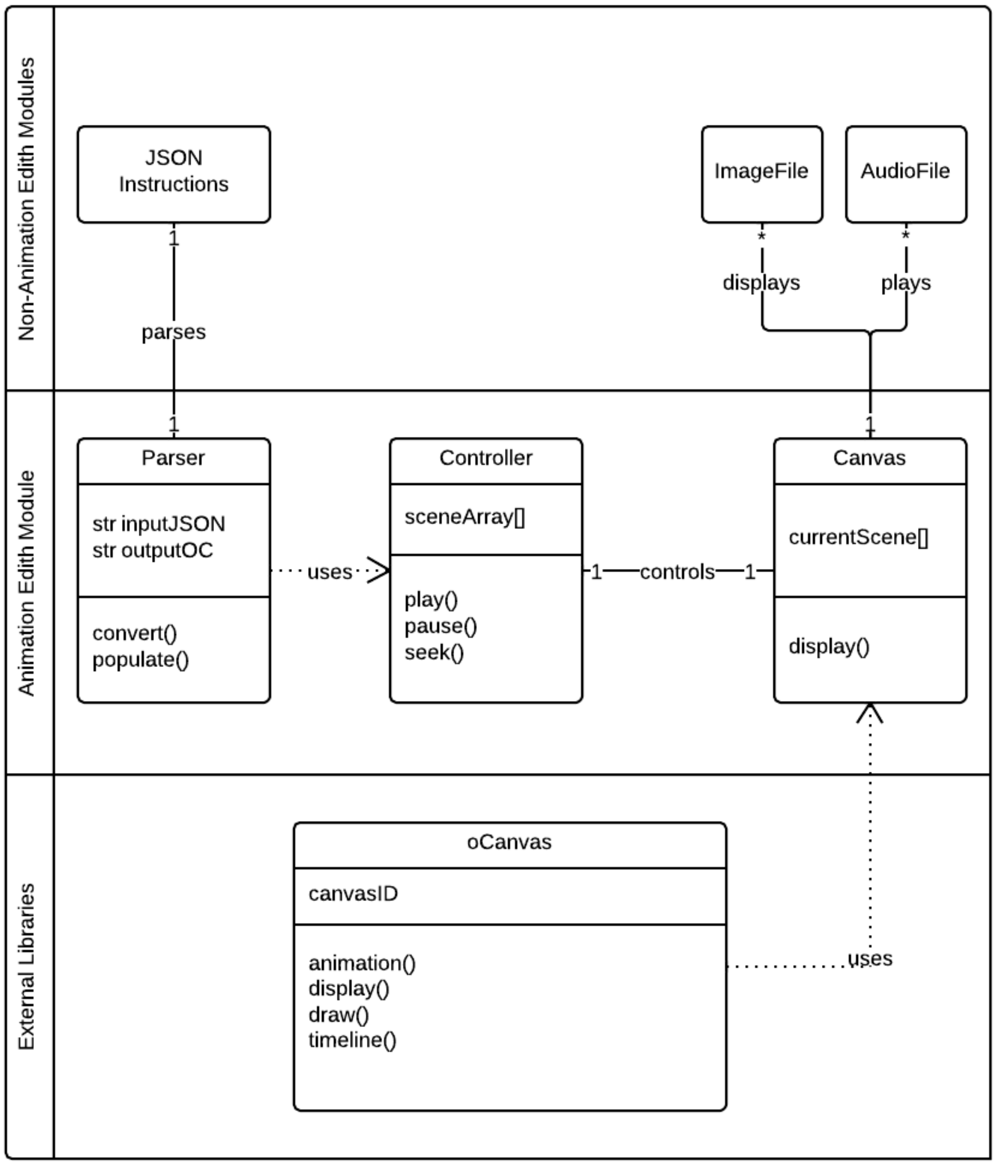
1.3 Interactions

(a) The Animation System will have no direct interaction with the "programmer" because the Animation System is a back-end module and is dependent on other frameworks and modules such as a canvas.

(b) The Animation System interacts with the Story Creator module, who feeds us the JSON instructions to be interpreted with the parser 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 better looking. 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

1.4 Activity Diagram



UML Class Diagram

2 *Project Status*

2.1 Current State of Implementation

- Our team has created an example HTML file which creates a HTML5 canvas with a sprite. This file implements the external HTML library, oCanvas which we use to bind several animations to the sprite. These animations can be invoked on the sprite by pressing the associated buttons. Even though this example simulates a final implementation of a scene, it provides a good example of how the sprite will react to certain animations in which we create.

  The animations we have implemented are rotate 360 degrees, vertical jump, forward jump and a reverse jump. Each animation also has an ease function associated that when applied, effects the playback of the animations to create different effects. For example, our jump animation looks best with a ease-in-out-bounce ease and a rotation looks best with a smoother easing function. Until we get a JSON file from the Story Creator we will continue to add animation elements on our example.

2.2 Member Responsibilities

Kramer:
Documentation with flexible responsibility between groups. Kramer will help where ever help is needed.

Zeke and Jon:
JSON parsing

Eric and Caluder:
Scene array and controller

3 *Glossary/References*

Glossary:

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

- Sprite: a computer graphic that may be moved on-screen and otherwise manipulated as a single entity (New Oxford American Dictionary (American English))