Edith

Animation System
Final Report

December 7, 2013

Eric Lund
Kramer Canfield
Zeke Rosenberg
Calder Whiteley
Jon Youmans

1. ***Project Summary***

1.1 Edith System

This should be about the edith system This should be about the edith system

1.2 Edith System

This should be about the animation system This should be about the anima-

tion system This should be about the animation system This should be about the animation system This should be about the animation system This should be about the animation system This should be about the animation system This should be about the animation system

2. *Development Procedures*

In this section, provide a brief discussion of your development process. Your goal is to explain your development and evaluate what worked and what didn't. You should be sure and address the following points:

What kind of process model did you follow? (Likely this is a hybrid of multiple models we discussed in class). What components did you work on in what order? What was effective and not effective about your process and how you structured your development? What forms of testing did you include? How do you know if your program works? What forms of testing were effective or ineffective? Which team members were responsible for which components? This should be more than "everyone worked on everything"– who contributed what to the system? Be sure and address all parts of the questions.

Include plenty of specifics in your descrption. This section should basically explain what you did in the course in terms of the final project, with enough detail that a reader could understand what activities you did and how you spent your time. Imagine writing for someone who hadn't taken the course and was curious what you did. A key part of this discussion is analyzing what worked and what didn't. Be sure and address why you think some part of the process model wasn't effecive for this project. For example: why or why didn't putting together a requirements document help? This section will likely be between 1 and 3 pages in length.

In this section, provide a brief discussion of your development process. Your goal is to explain your development and evaluate what worked and what didn't. You should be sure and address the following points:

What kind of process model did you follow? (Likely this is a hybrid of multiple models we discussed in class). What components did you work on in what order? What was effective and not effective about your process and how you structured your development? What forms of testing did you include? How do you know if your program works? What forms of testing were effective or ineffective? Which team members were responsible for which

components? This should be more than "everyone worked on everything"– who contributed what to the system? Be sure and address all parts of the questions.

Include plenty of specifics in your descrption. This section should basically explain what you did in the course in terms of the final project, with enough detail that a reader could understand what activities you did and how you spent your time. Imagine writing for someone who hadn't taken the course and was curious what you did. A key part of this discussion is analyzing what worked and what didn't. Be sure and address why you think some part of the process model wasn't effecive for this project. For example: why or why didn't putting together a requirements document help? This section will likely be between 1 and 3 pages in length.

In this section, provide a brief discussion of your development process. Your goal is to explain your development and evaluate what worked and what didn't. You should be sure and address the following points:

What kind of process model did you follow? (Likely this is a hybrid of multiple models we discussed in class). What components did you work on in what order? What was effective and not effective about your process and how you structured your development? What forms of testing did you include? How do you know if your program works? What forms of testing were effective or ineffective? Which team members were responsible for which components? This should be more than "everyone worked on everything"– who contributed what to the system? Be sure and address all parts of the questions.

Include plenty of specifics in your descrption. This section should basically explain what you did in the course in terms of the final project, with enough detail that a reader could understand what activities you did and how you spent your time. Imagine writing for someone who hadn't taken the course and was curious what you did. A key part of this discussion is analyzing what worked and what didn't. Be sure and address why you think some part of the process model wasn't effecive for this project. For example: why or why didn't putting together a requirements document help? This section will likely be between 1 and 3 pages in length.

In this section, provide a brief discussion of your development process. Your goal is to explain your development and evaluate what worked and what didn't. You should be sure and address the following points:

What kind of process model did you follow? (Likely this is a hybrid of multiple models we discussed in class). What components did you work on in what order? What was effective and not effective about your process and how you structured your development? What forms of testing did you include? How do you know if your program works? What forms of testing were effective or ineffective? Which team members were responsible for which components? This should be more than "everyone worked on everything"– who contributed what to the system? Be sure and address all parts of the questions.

Include plenty of specifics in your descrption. This section should basically explain what you did in the course in terms of the final project, with enough detail that a reader could understand what activities you did and how you spent your time. Imagine writing for someone who hadn't taken the course and was curious what you did. A key part of this discussion is analyzing what worked and what didn't. Be sure and address why you think some part of the process model wasn't effecive for this project. For example: why or why didn't putting together a requirements document help? This section will likely be between 1 and 3 pages in length.

3. *Requirements Evaluation*

Animation teams initial requirements contained an interesting combination of very specific and very vague requirements. This was not a huge problem for us, as we managed to change things around when we needed to in order to facilitate the same end effect that the initial requirements were going for. One of the primary functions of the Edith system is to teach object oriented programming skills, so we thought that it would be helpful if our animation system ran in an object oriented way.

Animation Requirements and Analysis

The Animation System would accept JSON instructions to be defined in an API that gave instructions for types of animations, durations and other parameters.

This requirement is one of the bigger ones that changed. There was some misunderstanding about how the final system would all work for awhile.

This requirement was written when our perception was that we would have our own canvas and possibly an HTML file with which to write in buttons and such for playback controls, etc. In the final Edith implementation, the animation canvas is just a canvas plopped onto the main page as a piece of the website. Thus, the actual HTML of the page was handled by the Story Creator team, and Animation provided our code solely via an external JavaScript file. There was then a second period of misunderstanding in which the requirements changed, and we were under the impression that this external JavaScript file would be called on by the Visual Editor team via hooks, so their blocks would call our code. Again, this changed, as in the final implementation we went with an altogether different version. Our final version contains a animationMain() method, which simply runs the JavaScript eval() function on a String of JavaScript code. This way, the Visual Editor team was able to construct the string using their blocks. These blocks simply added text to the string that was the aforementioned hooks to our program. The play button from the Story Creator team then simply called our main() method, with the parameter of the Visual Editors string of JavaScript code, which gets evaluated and runs like JavaScript. If there are any errors in the string, it will crash just as normal JavaScript would which is intended as we wanted to simulate a real programming experience and not something that worked every time. One thing that did end up as we said it would was the API that we created, which was incredibly helpful in creating the linkup between the visual editors panel and hooks to our code.

The JSON instructions would be read by the animation system, which would then create animations and sounds in a frame based system.

As mentioned above, we made the choice to use the oCanvas external library to ease the production of an object oriented experience for the user in the end. In the end, the use of the oCanvas turned out to be both a blessing and a curse. It definitely simplified things for us in the beginning, but as we looked towards some of our other requirements or even just tried to do additional things with our animations we were hampered by the system. The oCanvas uses animation queues for sprites to move around the queue. In regards to the sprites, we met our requirements, as that is how we described the animations to work. However, we had the notion that our animations would work because they would produce a series of frames, which we could easily store in an array, and then just flash onto the screen while iterating

through them. This did not end up working out, because the oCanvas does animations as a queue of actions with a length. If you want to do certain series of animations, it would actually be easier in a custom system. For example, if you want a sprite to begin moving across the screen, and then midway through the motion begin spinning as well, you run into problems. Calling move and then rotate will result in the sprite moving to the location, stopping, and then rotating. To get the desired animation you would need to start a new queue, and put the rotation in there. This still doesnt solve the issue, because now your sprite would start moving and spinning at the same time. So youd really have to make a new queue, put a wait() call in one, and then add the rotate to that. In our initial understanding, we thought that our methods would have some sort of a duration parameter as well as a timestamp parameter that told the system when it needed to be called. Because of the odd nature of the queue within the oCanvas, once we were overly reliant on that (which unfortunately came in the beginning of our development) it was very difficult to find work-arounds without sacrificing all of our previous work and starting over.

The Animation System provides video output to be taken by another group and drawn onto the canvas, simultaneously playing associated sound files in a standard audio format.

This requirement was half met. We did draw our animations onto a canvas that another group gave us a reference to, but it is not just a video file. Our animations work through an interactive display painted on a regular HTML 5 canvas. The downside of this is that there is not currently a way to export a video file created using Edith. However, using the framework is there for videos and projects to be fairly easily loaded by another user into Edith, although that functionality isnt implemented. There was a period of time when we considered adding audio to our videos, and our earlier integration demos actually featured a working video and audio combination, as well as an interface for easily associating sound files with a particular animation. However, there was no other group working on any of the other means that were needed for the audio to work, and they were all swamped with their development. What would have been needed: a way to share/access audio files via sharing framework, a way to upload your own sound file or access a stock database of files, and an interface for scrolling through given sounds and attaching them to animations, which in all would have required Shar-

ing Framework, Object Creator, and Visual Editor to all do additional work which was not specified at all in the requirements. Here is a good example of how we didnt really think through how we wanted the sound to work with the framework of the rest of the project. Obviously the sound worked within our animation system, as we had it working, but our requirements in the beginning did not make it clear what would need to happen from other groups.

The Animation System will have playback control functionality that will allow a user to pick a time to begin their video from, and watch it from their to enable easier testing and editing of videos within Edith

This requirement was not met. As described above, we ran into problems with the way that we chose to implement the object oriented nature of our sprite animations, specifically in oCanvas difficulty. Again, this was because we began implementation using the oCanvas without thinking through the process of how an array of frames for a scene might work. Once we got ahead of ourselves, it was very difficult to backtrack and find a solution for playback controls. We could have implemented a very rudimentary system that enabled the user to jump to specific instructions that they had chosen, or even to a specific time, but the outcome would have been quite messy. Problems that we anticipated were instructions that would effectively be lost if a user attempted to jump to a time that was part way between the start of the animation and the finish. This implementation would have been moderately useful for a user who wanted to jump to the end of a very long video, but we felt that it would have been buggy with no ways to fix the issues. Again, this was a product not only of our reliance on the oCanvas but also of a failure to comprehend how we would implement our requirements. In short, our requirements were initially too vague. In the end, we did not even implement a simple start/stop function. This is because the oCanvas has no way of stopping an animation and then resuming it, so we could stop animations, but then there was no other way to continue. We failed this requirement.

4. *System Design & Architecture*

Animation teams final implementation is contained within one javascript file and contains a list of functions that can be utilized by other groups. These functions are primarily functions that animate objects (sprites) on the canvas

but also include a few functions that allows for the ability to add sprites onto the canvas. To achieve this, our design utilizes and relies heavily on an external library called oCanvas. oCanvas is a JavaScript library that is intended to make development with a HTML5 Canvas easier. Instead of working with pixels, oCanvas allows for an object oriented interface allowing animations to be applied to objects, in our case, sprites. oCanvas can be thought of as strapping extra functionality onto a preexisting HTML5 canvas as all of the generic HTML5 canvas functions are still available and can be interweaved with any of the oCanvas functions. More information and documentation about oCanvas can be be found at their website: http://ocanvas.org. Because our code contained in one JavaScript file, we have appended the oCanvas library directly into our code as it can not be imported in line as it could be with HTML.

In order to begin using the animation functions the user needs to create a HTML5 canvas and assign it an ID. This ID can then be fed into our function called wrapHTMLCanvasToOcanvas(HTML5canvasID, background) which takes the HTML5 ID and a background color. This function simply takes the HTML5 canvas ID and calls the oCanvas function create() which creates the oCanvas. Our method then returns that oCanvas object. As stated above, this methods basically extends the HTML5 canvas adding more functionality to it – it does not actually create a new canvas. After attaining the oCanvas object, sprites can then be added by calling the function addSprite(theOCanvas, image, width, height, xcoord, ycoord). The first parameter to this method is the oCanvas object which is returned from the first function wrapHTMLCanvasToOcanvas. The image parameter is a string of an image path to the image to be represented as the sprite. The remaining parameters deal with the size and location of the sprite. Once those steps have been completed the user can then animate the sprite using any of our animation functions.

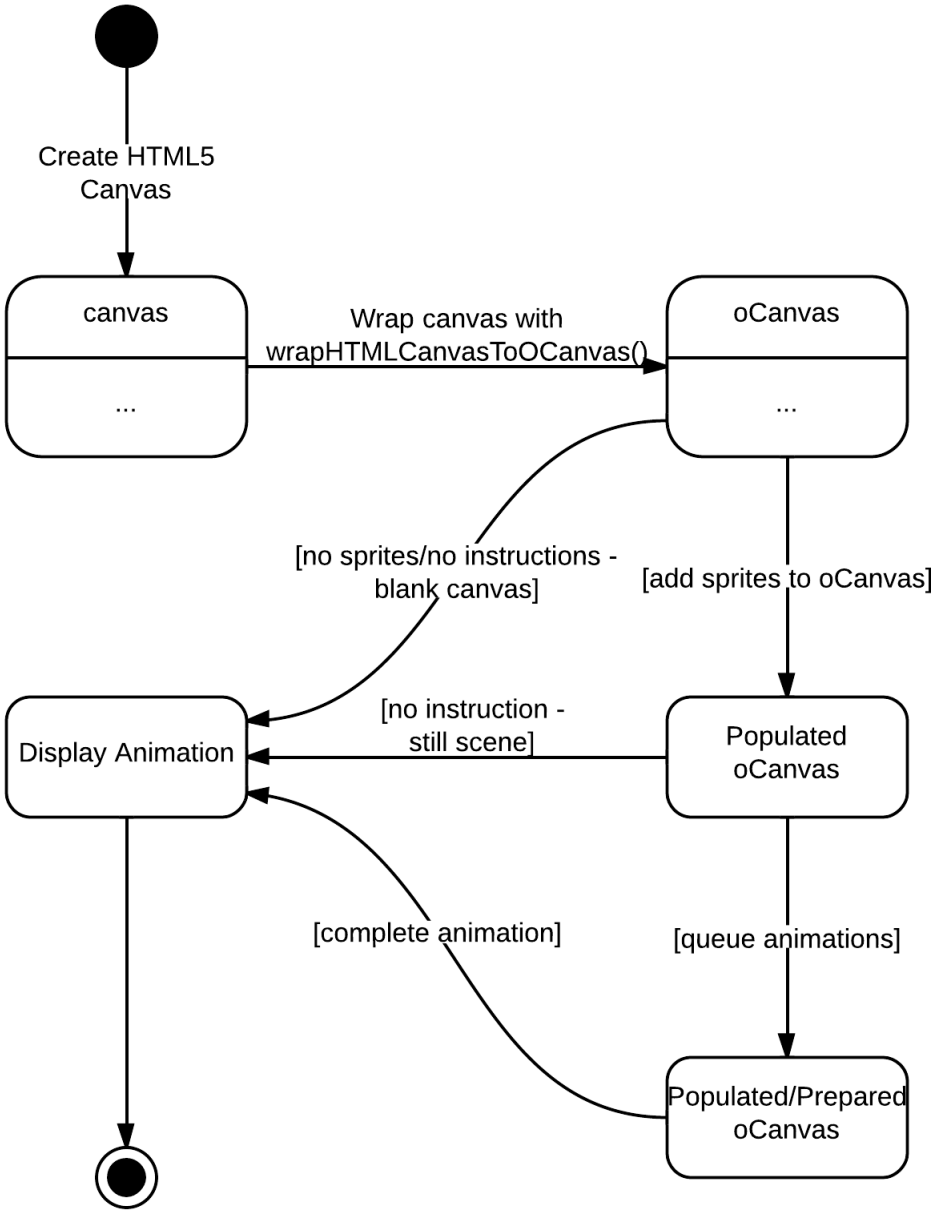
This process can be simplified to a series of 4 steps:

- Create HTML5 canvas assigning it an ID.
- wrapHTMLCanvasToOcanvas() to attain an oCanvas object.
- Add sprites by calling addSprite().

- Animate the sprites with the provided animation functions.

Below is a System state diagram that shows a sample implementation.

Figure 1. System State Diagram

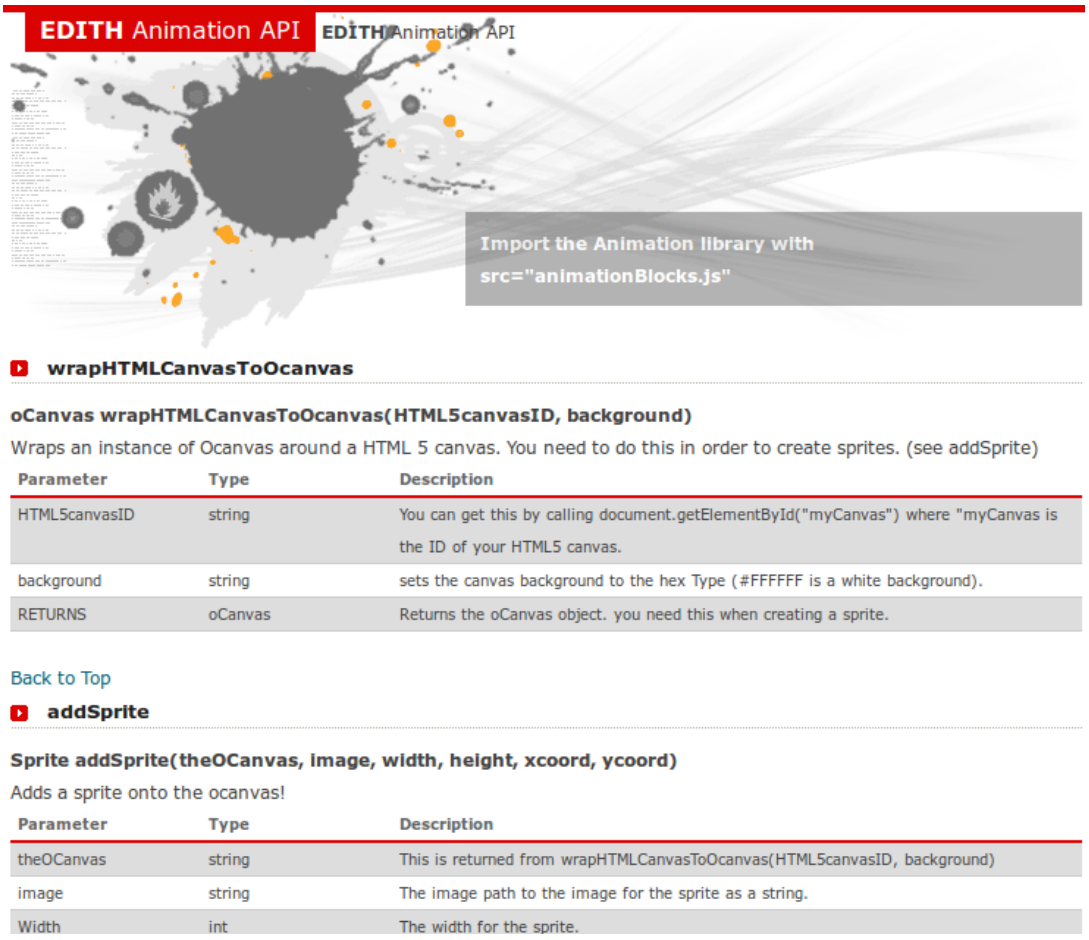


After creating an HTML5 canvas user calls the wrapHTMLCanvasToOCanvas() to create oCanvas object. At this point sprites and instructions can be added to be displayed on the canvas object.

Our animation functions include a generic animate function and a series of pre-defined animations created by our group. The generic function is called move() and can animate a sprite in any combination of the x-axis, y-axis, and z-axis (rotation). This function is intended to be used for simple animations but also allows the flexibility for the programmer to call a series of move functions to create a more complex animation. Our pre-defined animations include jump, double jump, move left, move right, move up, and move down which can be called in addition to the generic move. The main content behind all of the animation functions is a animation block provided by the oCanvas library. This animation block allows for many optional parameters that can be used to produce different effects. Some of the effects that we utilized are easing functions, duration time, and animation queues.

The animation system is conveniently documented in a clean browsable API. Each method is described in brief detail within the API site with all parameters listed and defined below. This documentation is intended for other developers who are using the animation system and provides all the details necessary to call and handle animation methods.

Figure 2. Animation System API



All animation methods are documented in an accessible API.

5. *Individual Reflections*

5.1 Eric Lund

Many challenges came up through development both technical and organizational throughout the whole development process. One of the biggest problems that stood out the most to me and falls in both the technical and organizational aspects was using GitHub for our source control. The idea behind Git is awesome, and should make the development process much easier and efficient when working with a group of people but I felt that in our case it held us back. It was a technical problem in that none of the members in our group had really worked with Git, or even a true source source control system prior to this project. There were multiple times in the beginning of development in which we actually lost some work (nothing substantial, luckily) due to us not knowing the technical aspects of how Git/GitHub worked. Though the development process we gained a lot of experience with works Git and I think it ended up being a helpful tool especially when we got to the integration periods with the other groups. The organizational challenge working with Git came by us not sufficiently managing our files in our branch. I was very hesitant (and I think I can speak for the team as well) in changing directories for files and moving folders around as it was only followed up by a series of merge conflicts and tracking problems. Because of that, our branch got messy real fast and wasn't as organized as it should of been. As a group it wasn't that big of a deal but if other groups wanted to look at what we had it would of been hard for them to find what they were looking for.

Our group utilized many software engineering techniques during the development process but what stood out to me the most was our communication abilities. Our group had several meetings in which all of our members were able to attend and stay up to speed with the current development process. We even had several meetings that we didn't do any coding but rather talked about how we were going about on our implantation and what our next steps were going to be. Reflecting back this really made the whole process much easier.

If I were to redo the whole process there are two things I would definitely

do. One of which would be to get a better grasp of Git/GitHub. I feel it would be beneficial if the whole group took time away from development to make sure we all knew and felt comfortable working with Git and also discuss as a group, how we would use it. Another thing I would do is communicate more with the other groups that directly affect what we are developing. As I said earlier, I felt the communication was exceptional in our group but it would of been even better if we extended that out to the other groups as well by staying updated on their current status and implementation. I had a great experience with my group members and felt like we accomplished the task at hand.

5.2 Kramer Canfield

KRAMER TYPES HERE

5.3 Zeke Rosenberg

ZEKE TYPES HERE

5.4 Calder Whiteley

CALDER TYPES HERE

5.5 Jon Youmans

Initially, I found the biggest challenge to be concretely defining the goal of our team. It was difficult to discuss and vet approaches to the problem at hand without being positively clear about the target. As the project continued the goal began to take shape and it became easier to see where the group's effort and focus belonged.

Organizationally, our team had a relaxed group structure that made it easy to contribute when our schedules conflicted. This allowed team members to manage their time effectively and balance their efforts for the project with demands from other classes.

While the group had a sound background in programming, as Eric said above, our version control experience was minimal at best and GitHub caused a lot of frustration for us. Its value is really clear but we continually marched forward focusing on development without stopping

to get truly comfortable with what git has to offer. Version control was a very valuable concept to me and possibly the most import thing I took away from this class. I intend to improve my understanding of git as I continue with software development.

LaTex was also a valuable technology to be introduced to. I plan to increase my familiarity with it going forward as well.

While our development style began with a large planning session our heading changed significantly throughout the project. This occurred most notably after integration meetings as the big picture became more firmly established. We designed our module to be somewhat flexible to changing requirements but the passive nature of the animation system, in the sense that it provides methods to be called by other systems, kept our team mostly insulated from sudden changes in requirements throughout the semester.

If we were to start this project again I would begin with more serious use cases and an immediate intergroup meeting to establish the group roles a little more clearly. Internally, I think we solved the problem quite efficiently by taking advantage of an existing animation library and tailoring it to the project. That being said, I think some of the reasoning behind our design comes from the pressure we had to firm up our implementation before other teams could specify the functionality they required.

6. *Glossary & References*