To: Chris Lindgren

From: Andrew Burke

Subject: DITA topic model design rationale

Date: 12/18/15

My DITA topic model details how to set up a basic video game using JavaScript and the <canvas> element. The topic model centers around three user tasks: preparing the <canvas> and JavaScript file, loading in an image and creating a moving character. The character creation task is a parent task to two sub-tasks: creating a player character and creating an enemy character. To surround these main tasks, I include a primer on the <canvas> element itself and its relation to video games, two reference topics discussing writing conventions in HTML and JavaScript and a concept topic explaining a <canvas> game’s update loop and what it looks like.

As for the architecture of the information, many of the tasks and topics in the model build off of one another. All of the core tasks involve writing JavaScript, so links to the reference topic on JavaScript are provided in all of them to give readers information they may need regardless of where in the topic model they begin reading. For a similar reason, the background image task and the character creation task also refer to the concept piece on the update loop, which is a critical code structure for game functionality. Those functions don’t work properly if the JavaScript is not set up to refresh the canvas, render images and modify values for each frame. Creating a character involves loading in an image in a very similar way to the process described in the background image task, so the character task refers to that task as a prerequisite before delivering its main content. The tasks for creating the player character and an enemy character then stem from that general character task, as they both require writing a JavaScript object containing numeric and Boolean values. While both sub-tasks involve modifying those values, the primary way that they differ is in how the JavaScript should go about that modifying: the player character should respond to user input on the keyboard, while an enemy character should follow a defined movement routine. The parent task of creating a character presents links to these two sub-tasks within a choice table, because of that similarity. There’s no need to write such similar steps twice for two separate tasks, and this setup recognizes that. I felt that my tasks all had similar prerequisites and knowledge needed to perform them, so I tried to draw relationships between my topics to reflect that.

I believe the relations between the topics of my model that I’ve drawn are effective in supporting my model’s main goal. Many of my tasks are similar in the knowledge a participant needs to execute them and how they are executed, so making my reference and concept topics available throughout all tasks is a strong choice. I combined reading in JavaScript and calling the <canvas> into one task because they both lead into the preparation of the video game. Creating the player character and creating an enemy character are also merged paired together for their similarity. I set up my tasks in an order where they can properly draw from each other’s information. The <canvas> and JavaScript file must be set up properly before images or characters can happen, so my preparation task is listed before the others in the model. The character creation task requires an image to be loaded in, so it comes after the image loading task to avoid referencing it out of turn. This structure, I believe, properly demonstrates the knowledge needed to build a video game in <canvas>.

I made many design decisions over the course of designing and writing my DITA topic model that led me to the model I have now. Right out of the gate, I had to decide what kind of scope I wanted the model to have, with two different possibilities: general functions and tasks of the <canvas> element or the functions and tasks that were specific to video game functionality. The two paths shared the task of displaying a background image, but my preliminary design for a topic model on general <canvas> work also had tasks for writing the update loop and enabling user input. My struggle to think of a worthy concept topic for this model led me to believe that the video game-specific topic model might work better for me; I would be able to use the main tasks of the general model as concept and reference topics while keeping background image loading as a task and creating two more tasks out of creating the player character and an enemy character. I also liked this model because of the opportunities for reusability. For instance, both the background and a character use the same image loading functions to grab images, so I could use one task to support another. Believing that the more specific topic model fit the requirements of the project better, I decided to adopt that one over the more general model.

Once I committed to my video game-based model, my next major decision was to determine what tasks to cover. At first I was going to use the three central practices I researched for my historicizing: how to draw the background, how to make the player character jump and how to add an enemy character. I decided to expand the scope from how to make the player character jump to simply how to make the player character, and I then I encountered the major roadblock for my tasks. As I found out, creating the player character and creating an enemy character were very similar tasks. I thought it wouldn’t be proper to write two major tasks that followed almost the same steps to completion, so I switched out the enemy character task for one on how to write collision detection. I abandoned this idea after reading and learning more about how DITA worked and what it can do. I learned that I could rewrite my current player character task to just describe how to create a character, and then link to two different sub-tasks for the player character and the enemy character using a choice table element. This method let me stick to my original plan for this project and make smarter use of DITA tools, so it seemed to me the best decision to make.

The choice on how to scope my topic model and how to approach the enemy character task stood out as the two major design decisions I had to make as I wrote my project. Other minor decisions and questions arose as I worked. After settling on a character creation task, I felt the need to rewrite my background image topic to be only about loading images. I wanted to refer to this topic as an initial step for my new character creation task, so I didn’t think it should use language specific to making backgrounds anymore. Originally, I had two smaller concept topics on how a <canvas> element is created in HTML and called in JavaScript and how an HTML document reads in a JavaScript file. As I worked on these, I couldn’t envision them properly as concepts, and they both seemed to be short topics that didn’t cover much information on their own. To that end, I chose to combine the two and write them as one of my major task topics, with loading images and creating characters. This task fit in as a first step for the loading images task the same way the loading images task was a first step for the character creation task. I already had a reference topic on JavaScript writing conventions at this point, but my new <canvas> preparation task led me to realize I may need a companion reference topic on HTML writing conventions. Deciding it was necessary, I wrote a second reference topic about the tags and elements a participant would need to write in HTML to set up their <canvas> properly.