**Specification Document for Towers of Hanoi**

**System Description**

* **Give an overview and functional description of the system. Provide enough detail so that the reader understands what the project is and what it is to accomplish.**

[Clayton]: The system under development will be used as an interactive visual representation of the sequence of moves performed to solve the Towers of Hanoi.

<https://en.wikipedia.org/wiki/Tower_of_Hanoi>

* **Include a description of the user interface.**

[Clayton]: The end-user will be displayed with the 3 rods (source, destination, auxiliary) and the current N disks stacked on each of the 3 rods from the Towers of Hanoi problem. The end-user is allowed to move the disks step-by-step in the forward or reverse sequence of moves to solve the Towers of Hanoi. The end-user is allowed to choose the quantity N disks ( N should be a small enough to visually fit N disks on the display and N should also be small enough so the problem can be solved within a reasonable amount of time ).

* **Write the description from a user's point of view—describe what you're going to design.**

[Clayton]: Each of the disks can be moved forward or backward one step at a time based on user-input ( user-input: **next** or **prev** ), or the sequence of moves can be played automatically based on a certain wait amount of time between automatic moves ( user-input: **play forward**, **play reverse**, **pause** ).

**Test Requirements**

* Write a system test plan—when testing the completed system, *what* needs to be tested?

[Clayton]: The following should be tested:

1. the sequence of moves performed to solve the Towers of Hanoi problem
   1. only move one disk at a time
   2. only the top-most disk can only be moved to another rod
   3. only smaller disks can be placed on top of larger disks
   4. start with N disks on source rod
   5. end with N disks on destination rod
2. the forward / reverse step-by-step functionality from user-input
3. the pause / play forward / play reverse functionality from user-input

* **Do validation or acceptance testing to make sure that you meet the user's requirements. You will not have time to field-test your project. If you were to do so, however, how would you accomplish this testing?**

[Clayton]: Field-test is irrelevant for my project. I will plan to demonstrate my project to friends / colleagues. Based on their usage and understanding of my project, I will subjectively consider my project a success or failure. Success will be achieved if Towers of Hanoi problem, interface, and solution is intuitive for simple end-user understanding and enjoyment. Failure will be declared if the problem, interface, or solution cannot be easily understood or any part of my project causes an unpleasant end-user experience.

* Do verification—internal tests, like module tests—of the components of your project; this is not a test of the complete system. For example, test the boundary conditions in your class member functions.

[Clayton]: Use the C++ unit lite test framework to verify the state of the disks on the rods is proper such that there are N disks, each of those N disks are on top of only larger disks, and each of the N disks is on exactly one of the 3 rods.

**Initial Design Strategy**

* Include your initial thoughts on how you will implement your project.

[Clayton]: I will plan to first implement this project using a command line interface (CLI), and then add a user interface (UI) to replace the CLI. In order to perform this incremental development in a modular fashion, I will plan to use the model-view-controller (MVC) design pattern. So in theory, I should only need to change the view and controller to use a UI instead of CLI. The model should remain unchanged.

**Design Document**

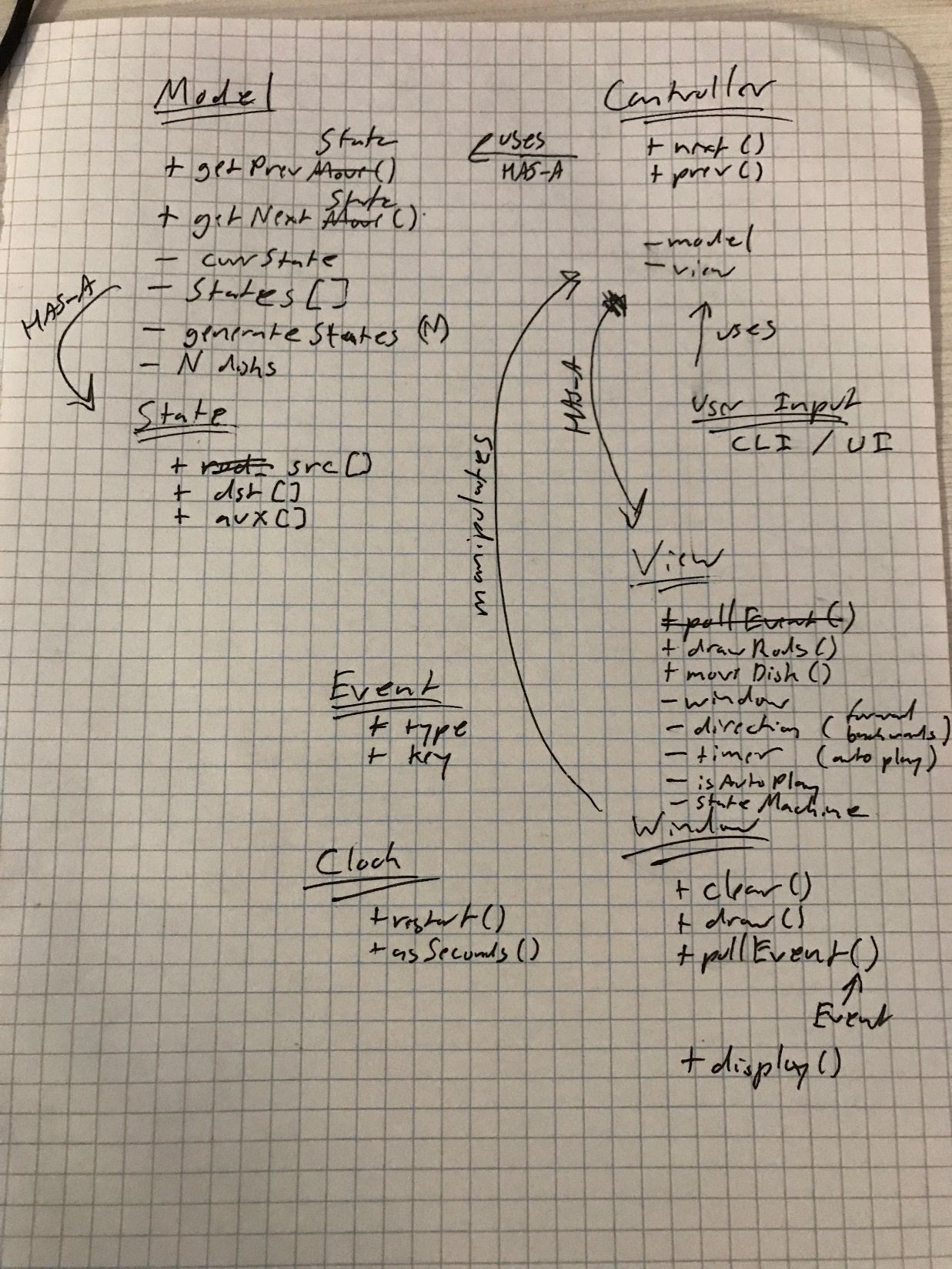
1. Give an overview of the class hierarchies. Remember the first class when we talked about design methodologies. I recommend using CRC cards for object discovery.

The Controller has-a Model which is manipulated by the end-user through the View. The View uses end-user input to provide feedback to the controller based on event polling from the Window. Event and Clock are used to receive input and update the views.

1. Discuss why a particular design was chosen.

[Clayton]: MVC design was chosen in order to decouple the view/controller from the model, since I am planning to start development with CLI-based view/controller and end with UI-based view/controller.

1. Include class diagrams and other documentation to describe your design.



1. Include public interface for each class.



This document should be written from a designer's point of view. It should describe *how* you're going to implement the design and should give a road map of **how** the components of your design fit together.