CSC258 Project Proposal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | First Name | Last Name | Student No. | Email |
| Member 1 | Ming Feng | Wan | 1002531236 | ming.wan@mail.utoronto.ca |
| Member 2 | Hao | Yu | 1004420021 | hpeiyu.yu@mail.utoronto.ca |

What is the title of your project?

Learning Rock Paper Scissors

Provide a one-paragraph description of your project.

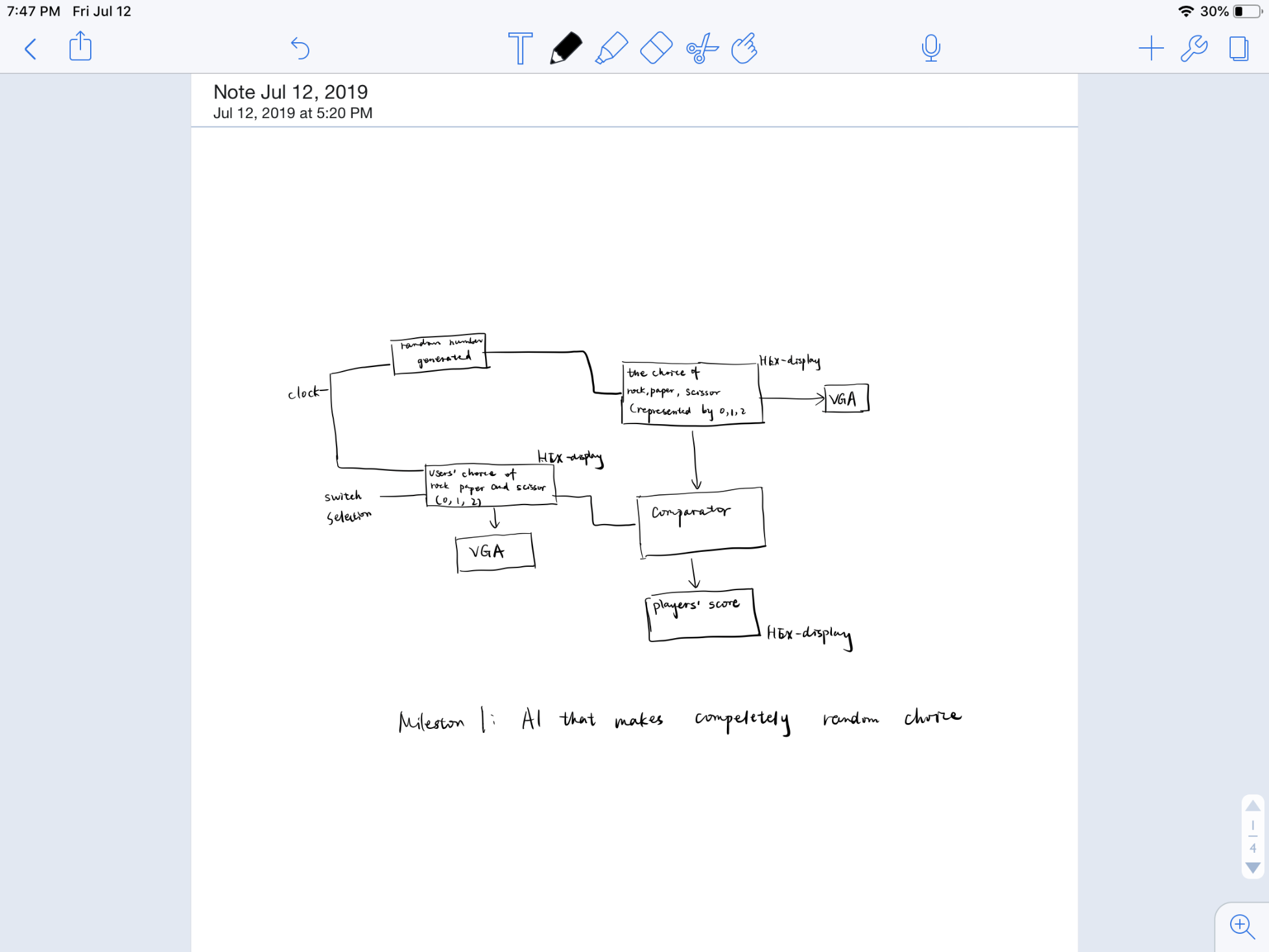
Wining a consecutive Rock Paper Scissors game requires a player to be unpredictable. Unpredictability is achieved by making completely random actions. For a human player, however, randomness in actions is difficult to achieve. Therefore, an AI can learn to predict a human player’s move in a Rock Paper Scissors game by “learning” the patterns of the human player’s action.

Project Description:

(This is where you describe your project in detail. You can use the Design Case Studies slides as a reference on how to create the following components for your project. All these components are not compulsory for you to have but most projects usually have these components. They are: high level pseudo code, state diagram, datapath-and-control-CU block diagram, input/output block diagram [Example: <https://www.nandland.com/goboard/images/project10-pong-block-diagram.png>], truth tables etc. These components should be designed and described to show your understanding of your design i.e. how many bits is each input/output, what is the max number your counter can count up to, how many counters/shift registers you need to use etc. A good idea is to get an initial draft of this done and show to your TA in the next lab or during office hours to get feedback.)

The human player can choose to play against three different AIs:

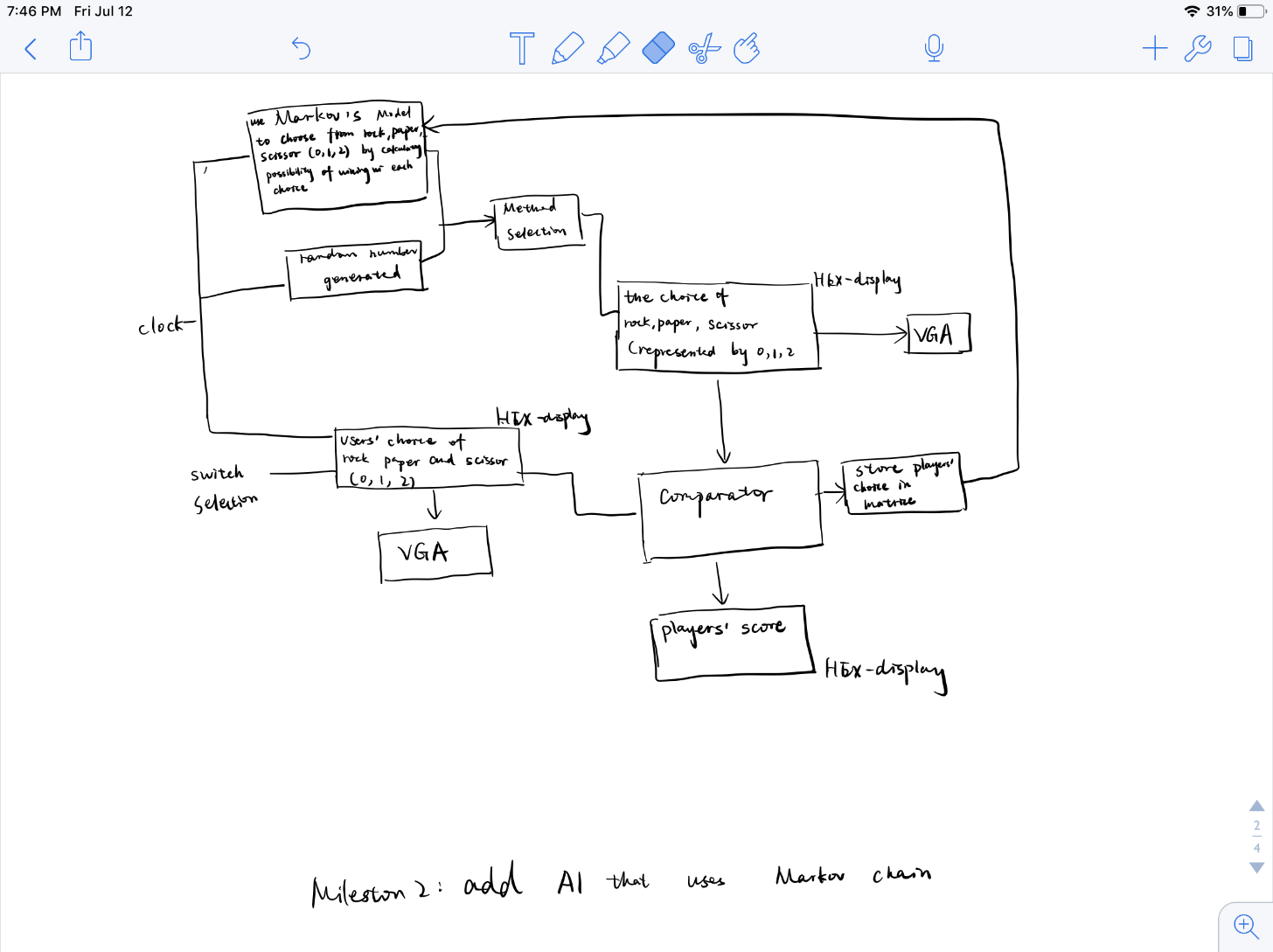
1. The first AI makes completely random actions, which can be achieved by implementing a Linear-feedback shift register.



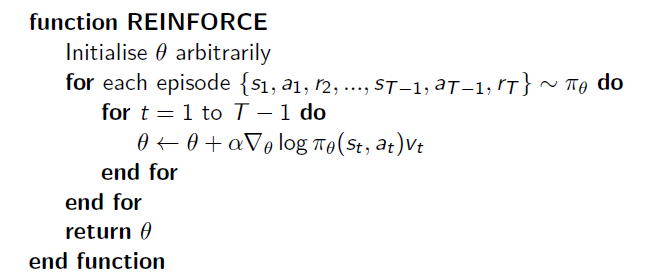
1. The second AI implements a discrete-time Markov chain. The AI gradually learns the player’s action pattern by updating the conditional probability of an action based on the given state.

For example, we assume that in the first round, the AI plays rock, and the human player plays paper. In the second round, the human player plays scissors:

There are nine states in total, and each state can lead to three possible actions. Therefore, implementation of this AI requires matrices, which can be presented as real registers, to record the conditional probabilities of action based on given state. A random number generator is also required.

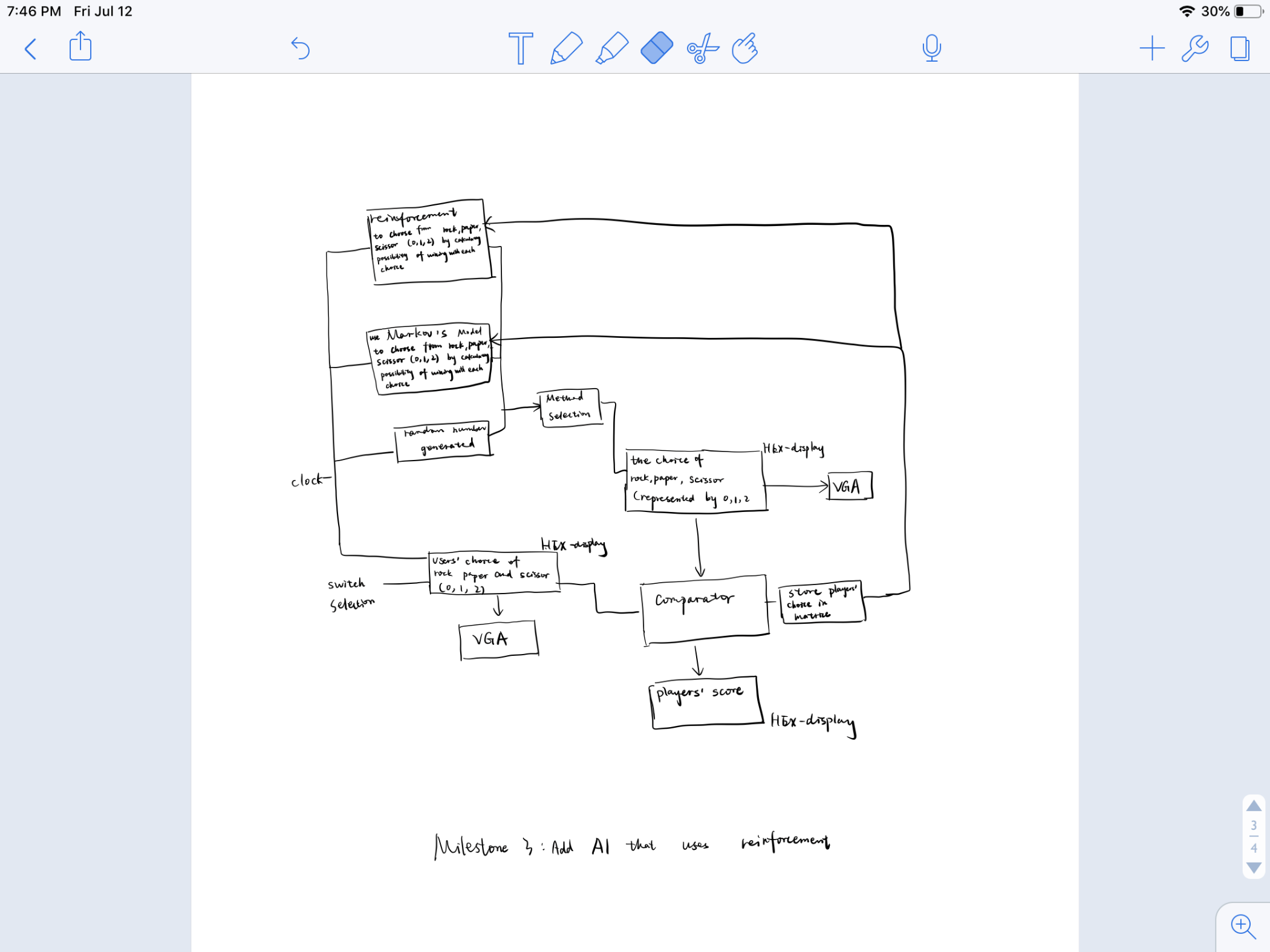


1. The third AI uses the REINFORCE algorithm (also known as Monte Carlo Policy Gradient). ([Picture credit](http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html))



We will use softmax policy for our AI.

We need real registers to store matrix values. A random number generation is also required to initialise theta.



What will you accomplish for the first milestone?

(Advice here: Pretend that you're designing Lab 8 around your project idea, in keeping with the difficulty level of the previous labs. Try to be **specific and detailed** in describing the components that you will complete. Don't say that you'll "think about" or "plan" or "design" something.

Bad example 1: We build the graphical interface.

Good example 1: We display moving spaceships on VGA.

Bad example 2: We write code for the PS2 keyboard interface.

Good example 2: We make the PS2 keyboard work and show the key inputs on the HEX display.

Assume your project can be developed in three independent parts, what you write in the space below should outline the components of the first part. Make sure to describe a full lab's worth of work, including the evidence of your work that you will provide to the TAs to justify getting the full marks for this milestone.)

Display the gaming interface with FPGA keys used for the player to select rock, paper, and scissors. Use switches for reset and select AI and use HEX display as score recorder.

Show that the first AI work. i.e. it can make completely random choice among rock, paper, scissors. The win rate of the AI against a human player should be around fifty percent.

What will you accomplish for the second milestone?

(similar advice as above, but for the second part of your project. Remember to specify what inputs and outputs will be used for each milestone. If your project is a visual game for example, what will appear on the screen for each milestone, for example, static colored boxes in one milestone and moving boxes in the next one etc.)

Implement Markov’s model and use scores to demonstrates that the second AI will win a human player most of the time if the player’s action follows a fixed pattern.

What will you accomplish for the third milestone?

(don't say "everything" just because this is the final milestone; describe the final components instead, and exactly what the TAs should expect to see)

Option1: Make different sound for winning and losing the game. Implement the REINFORCE algorithm and show that the third AI can win human players if the player’s action follows a fixed pattern.

Option2 : Indicate player’s and computer’s selection among paper, scissors, or rock on the screen by displaying corresponding graphics.

How does this project relate to material covered in CSC258?

FPGA board

Verilog code

VGA adapter

Counter

Clock

Mux

Hex display

Registers

What's cool about this project (to CSC258 students and non-CSC258 students)?

For a human player, the outcome of a Rock Paper Scissors game may be purely dependent on luck. We are trying to make our AI outperform human by attempting to win with the help of simple Machine Learning algorithms.

Why does the idea of working on this appeal to you personally?

1. I came across multiple Rock-Paper-Scissor AI built with discrete Markov chain, but I have never seen any Rock-Paper-Scissor AI built with Monte Carlo Policy Gradient. Therefore, I have great curiosity regarding whether REINFORCE algorithm with softmax policy would outperform discrete Markov chain, since the REINFORCE algorithm itself is based on Markov chain.
2. Implementation of Machine Learning algorithms generally demands high processing power, and therefore building a simple game with quick and responsive Reinforcement Learning AI often requires expensive computing hardware. Combing FPGA board and Verilog code with Machine Learning algorithms may be an effective and convenient way to significantly decrease the cost of computing hardware.