QLearning For Videogames

Steven Walton
University of Oregon

CIS 510: Multi-Agent Systems
3 June 2019

What is Q-Learning?

- Markov Decision Processes (MDPs) can represent decision making processes with random outcomes.

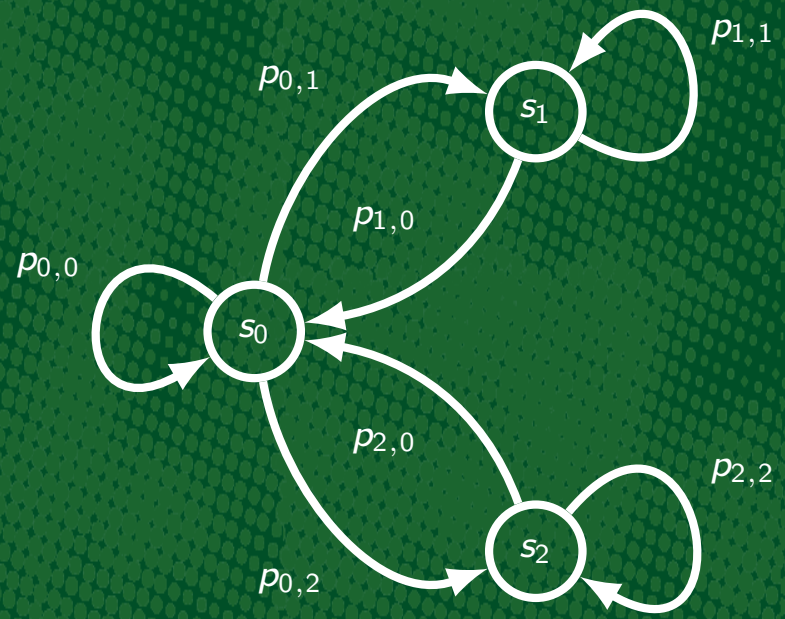


Figure: Sample Markov Decision Process

What is Q-Learning?

- Markov Decision Processes (MDPs) can represent decision making processes with random outcomes.
- We want to find an optimal policy (set of actions)

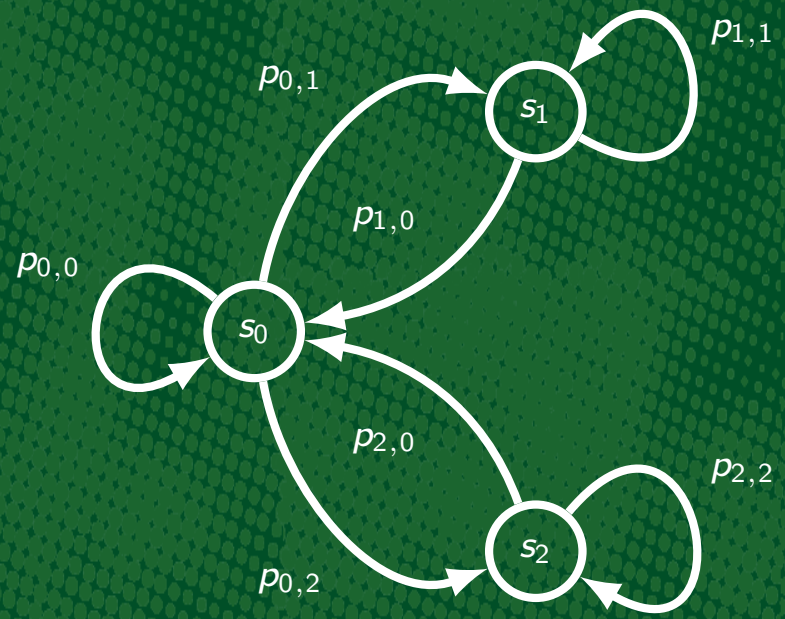


Figure: Sample Markov Decision Process

What is Q-Learning?

- Markov Decision Processes (MDPs) can represent decision making processes with random outcomes.
- We want to find an optimal policy (set of actions)
- We learn with the Bellman Equation, using an iterative process

$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$

$\alpha =$ learning rate

$r =$ reward

$\gamma =$ discount

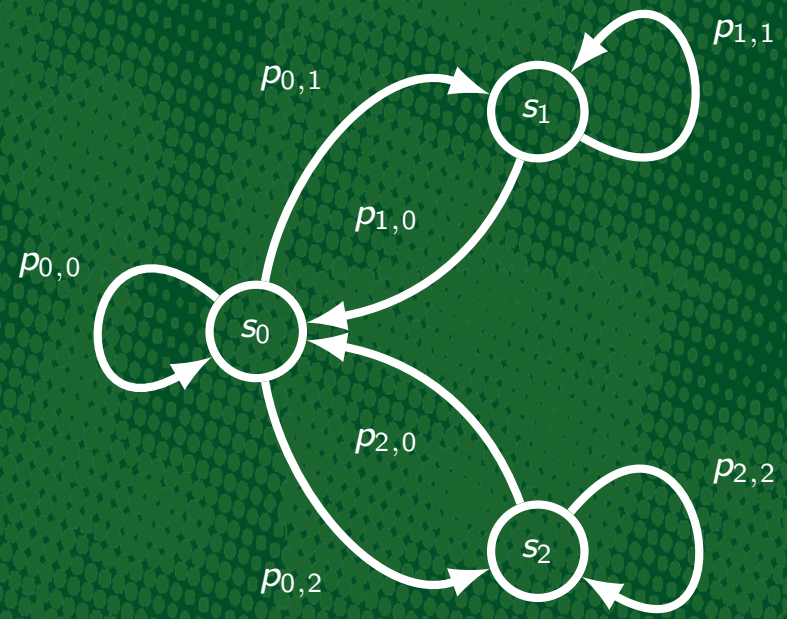


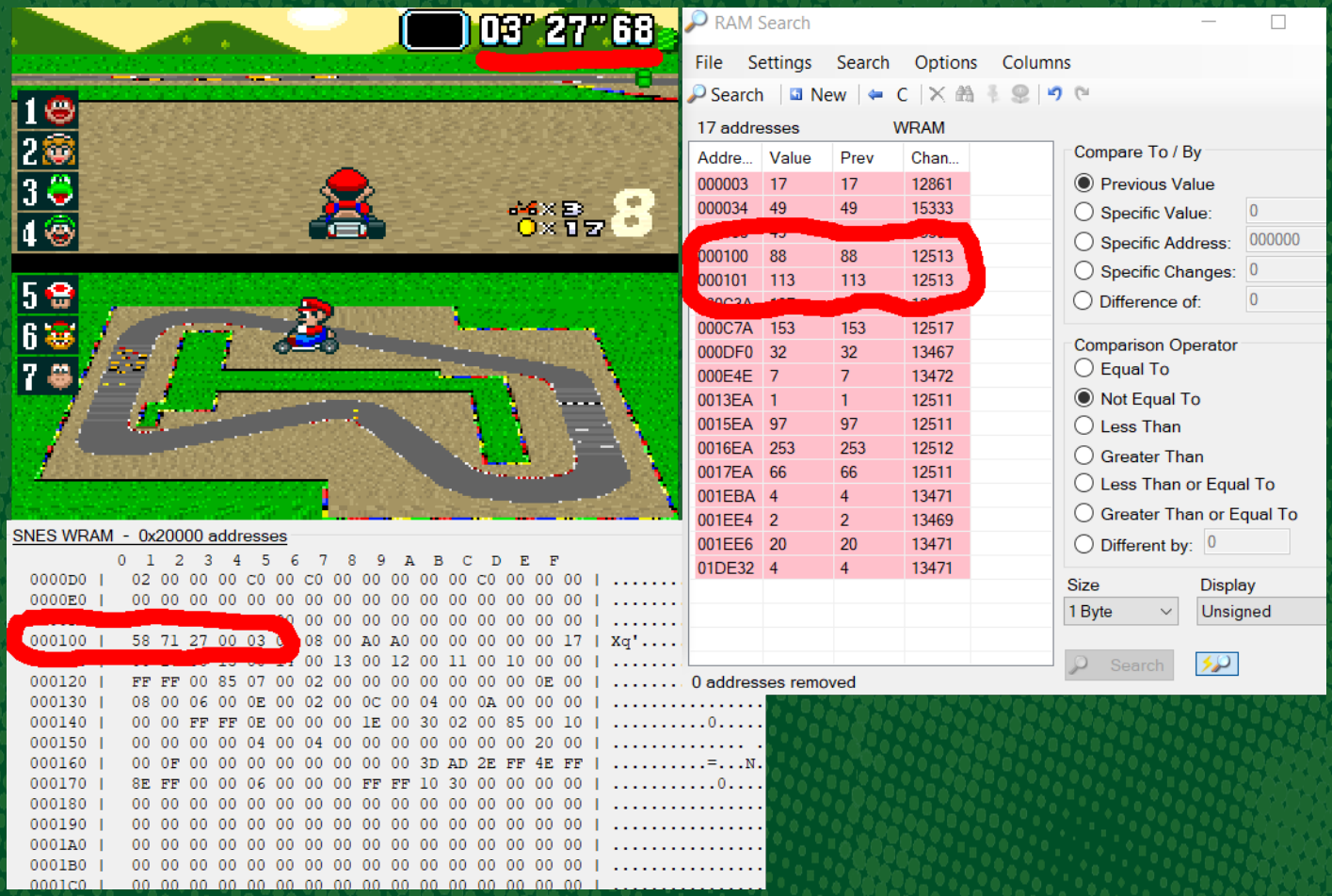
Figure: Sample Markov Decision Process

Framework

- OpenAI Gym: Allows for abstracts out environments and allows researchers to focus on creating algorithms.
- OpenAI Retro: Wrapps Gym framework to focus on retro videogames.
- Allows same code to be run on any game that is integrated.
- Creates an environment to (supposedly) allow for easy integration of new games

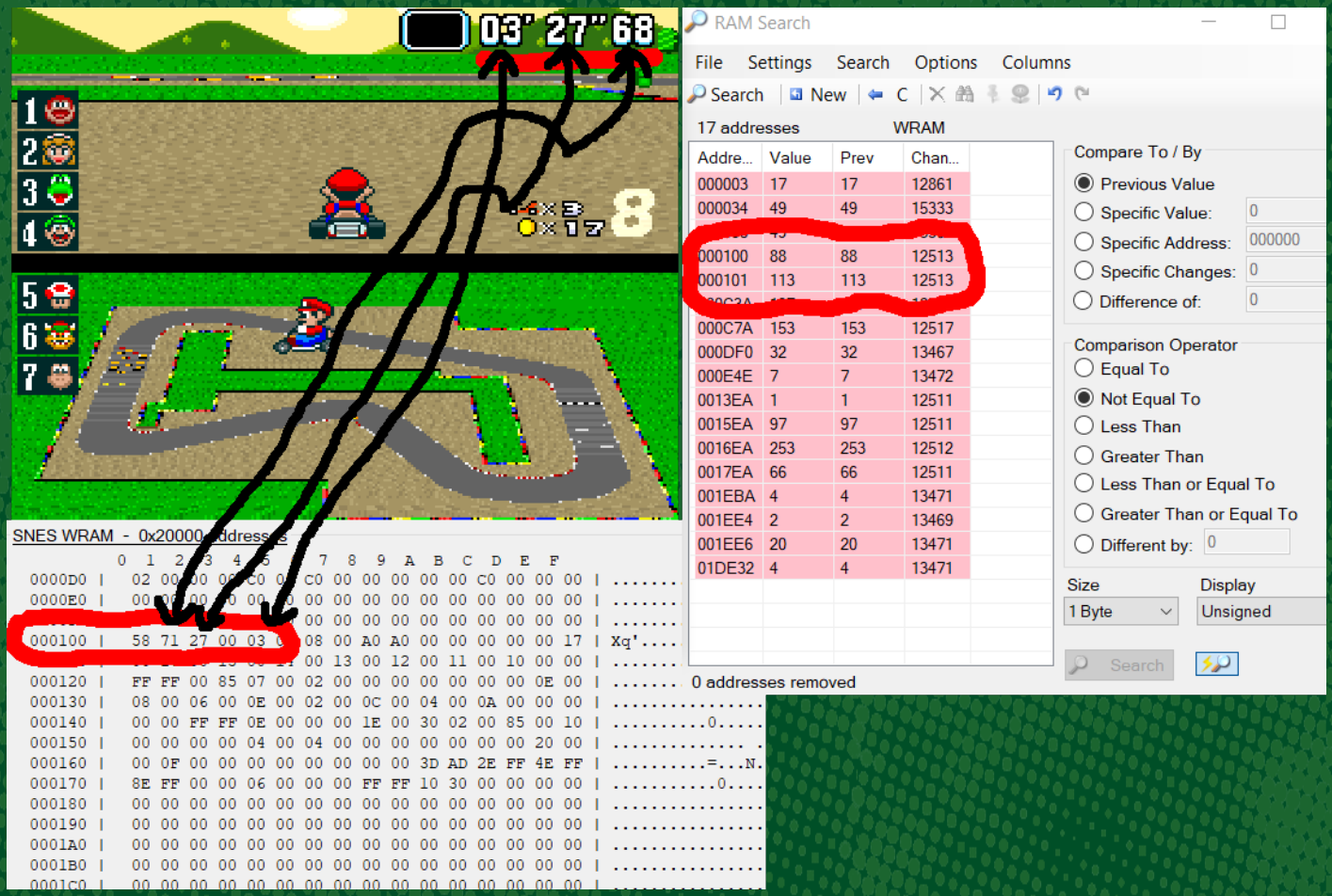
Integration

- Need to collect RAM values (used BizHawk).



Integration

- Need to collect RAM values (used BizHawk).



Integration

- Need to collect RAM values (used BizHawk).
- Convert Hex to decimal and add rambase number and integrate into scenario and data files

```
{
    "info": {
        "lap": {
            "address": 8261825,
            "type": "|u1"
        },
        "lapsize": {
            "address": 8257864,
            "type": "|u1"
        },
        "checkpoint": {
            "address": 8261852,
            "type": "|u1"
        },
        "rank": {
            "address": 8261696,
            "type": "|u1"
        },
        "coins": {
            "address": 8261120,
            "type": "|u1"
        },
        "minute": {
            "address": 8257796,
            "type": "|u1"
        },
        "second": {
            "address": 8257794,
            "type": |u1"
        },
        "milisecond": {
            "address": 8257793,
            "type": |u1"
        }
    }
}
```

```
{
    "done": {
        "variables": {
            "finish": {
                "reference": "coins",
                "op": "eq"
            }
        }
    },
    "reward": {
        "variables": {
            "rank": {
                "reward": 1.0
            }
        }
    },
    "actions": [
        [[], ["UP"], ["DOWN"]],
        [[], ["LEFT"], ["RIGHT"]],
        [[], ["A"], ["B"], ["Y"], ["START"], ["R"], ["R", "A"], ["R", "B"], ["R", "Y"], ["R", "A", "B"], ["R", "A", "Y"], ["A", "B"], ["A", "Y"]]
}
```

Integration

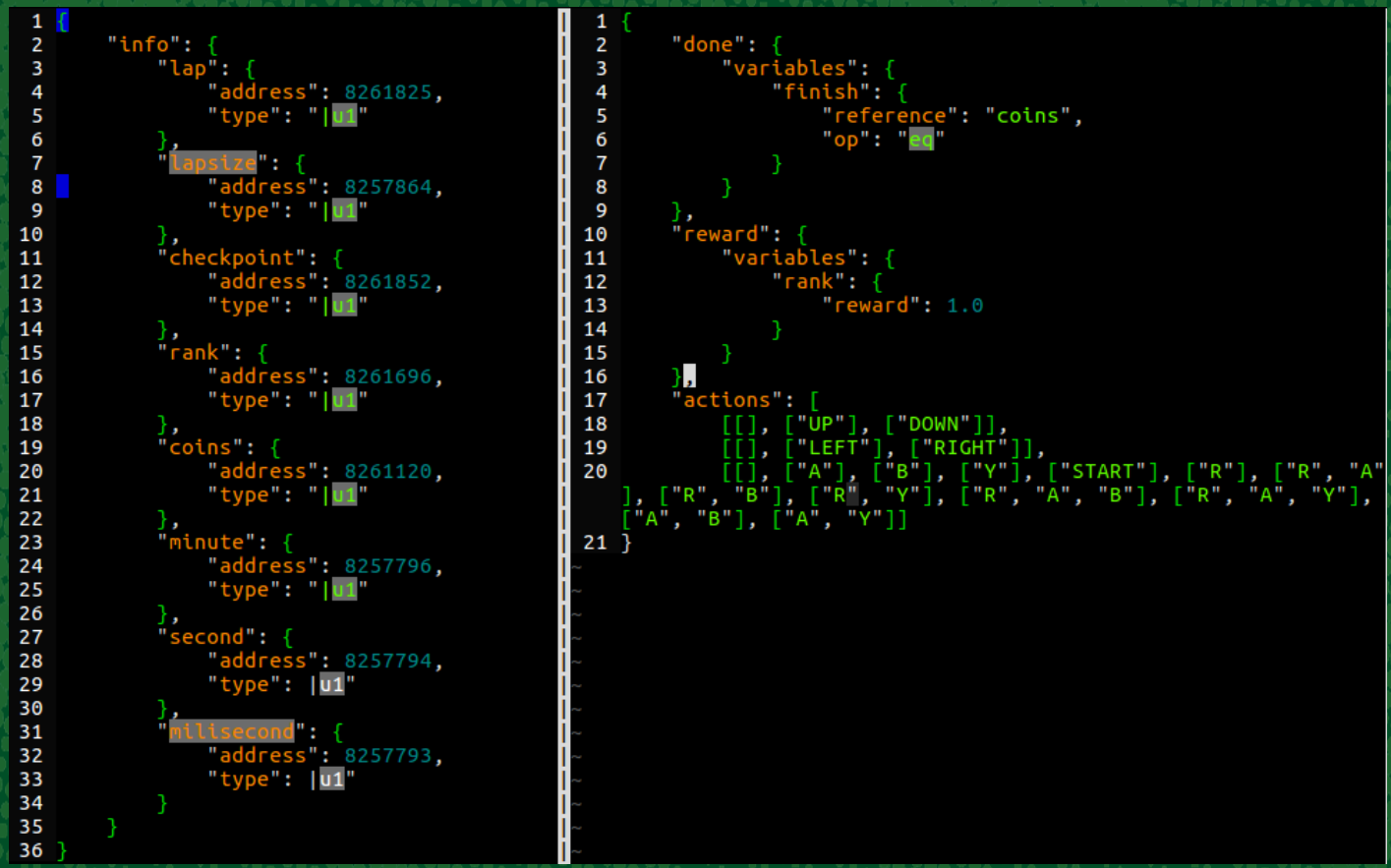
- Need to collect RAM values (used BizHawk).
- Convert Hex to decimal and add rambase number and integrate into scenario and data files
- Define actions and rewards for the game.

```
{
    "info": {
        "lap": {
            "address": 8261825,
            "type": "|u1"
        },
        "lapsize": {
            "address": 8257864,
            "type": "|u1"
        },
        "checkpoint": {
            "address": 8261852,
            "type": "|u1"
        },
        "rank": {
            "address": 8261696,
            "type": "|u1"
        },
        "coins": {
            "address": 8261120,
            "type": "|u1"
        },
        "minute": {
            "address": 8257796,
            "type": "|u1"
        },
        "second": {
            "address": 8257794,
            "type": |u1"
        },
        "milisecond": {
            "address": 8257793,
            "type": |u1"
        }
    }
}
```

```
{
    "done": {
        "variables": {
            "finish": {
                "reference": "coins",
                "op": "eq"
            }
        }
    },
    "reward": {
        "variables": {
            "rank": {
                "reward": 1.0
            }
        }
    },
    "actions": [
        [[], ["UP"], ["DOWN"]],
        [[], ["LEFT"], ["RIGHT"]],
        [[], ["A"], ["B"], ["Y"], ["START"], ["R"], ["R", "A"], ["R", "B"], ["R", "Y"], ["R", "A", "B"], ["R", "A", "Y"], ["A", "B"], ["A", "Y"]]
}
```

Integration

- Need to collect RAM values (used BizHawk).
- Convert Hex to decimal and add rambase number and integrate into scenario and data files
- Define actions and rewards for the game.
- Not all RAM values are so easy to get nor define. (Lua scripts)



```
{
    "info": {
        "lap": {
            "address": 8261825,
            "type": "|u1"
        },
        "lapsize": {
            "address": 8257864,
            "type": "|u1"
        },
        "checkpoint": {
            "address": 8261852,
            "type": "|u1"
        },
        "rank": {
            "address": 8261696,
            "type": "|u1"
        },
        "coins": {
            "address": 8261120,
            "type": "|u1"
        },
        "minute": {
            "address": 8257796,
            "type": "|u1"
        },
        "second": {
            "address": 8257794,
            "type": |u1"
        },
        "milisecond": {
            "address": 8257793,
            "type": |u1"
        }
    }
}
```

```
{
    "done": {
        "variables": {
            "finish": {
                "reference": "coins",
                "op": "eq"
            }
        }
    },
    "reward": {
        "variables": {
            "rank": {
                "reward": 1.0
            }
        }
    },
    "actions": [
        [[], ["UP"], ["DOWN"]],
        [[], ["LEFT"], ["RIGHT"]],
        [[], ["A"], ["B"], ["Y"], ["START"], ["R"], ["R", "A"], ["R", "B"], ["R", "Y"], ["R", "A", "B"], ["R", "A", "Y"], ["A", "B"], ["A", "Y"]]
}
```

Integration

- Need to collect RAM values (used BizHawk).
- Convert Hex to decimal and add rambase number and integrate into scenario and data files
- Define actions and rewards for the game.
- Not all RAM values are so easy to get nor define. (Lua scripts)
- Thanks SethBling

```
{
    "info": {
        "lap": {
            "address": 8261825,
            "type": "|u1"
        },
        "lapsize": {
            "address": 8257864,
            "type": "|u1"
        },
        "checkpoint": {
            "address": 8261852,
            "type": "|u1"
        },
        "rank": {
            "address": 8261696,
            "type": "|u1"
        },
        "coins": {
            "address": 8261120,
            "type": "|u1"
        },
        "minute": {
            "address": 8257796,
            "type": "|u1"
        },
        "second": {
            "address": 8257794,
            "type": |u1"
        },
        "milisecond": {
            "address": 8257793,
            "type": |u1"
        }
    }
}
```

```
{
    "done": {
        "variables": {
            "finish": {
                "reference": "coins",
                "op": "eq"
            }
        }
    },
    "reward": {
        "variables": {
            "rank": {
                "reward": 1.0
            }
        }
    },
    "actions": [
        [[], ["UP"], ["DOWN"]],
        [[], ["LEFT"], ["RIGHT"]],
        [[], ["A"], ["B"], ["Y"], ["START"], ["R"], ["R", "A"], ["R", "B"], ["R", "Y"], ["R", "A", "B"], ["R", "A", "Y"], ["A", "B"], ["A", "Y"]]
}
```

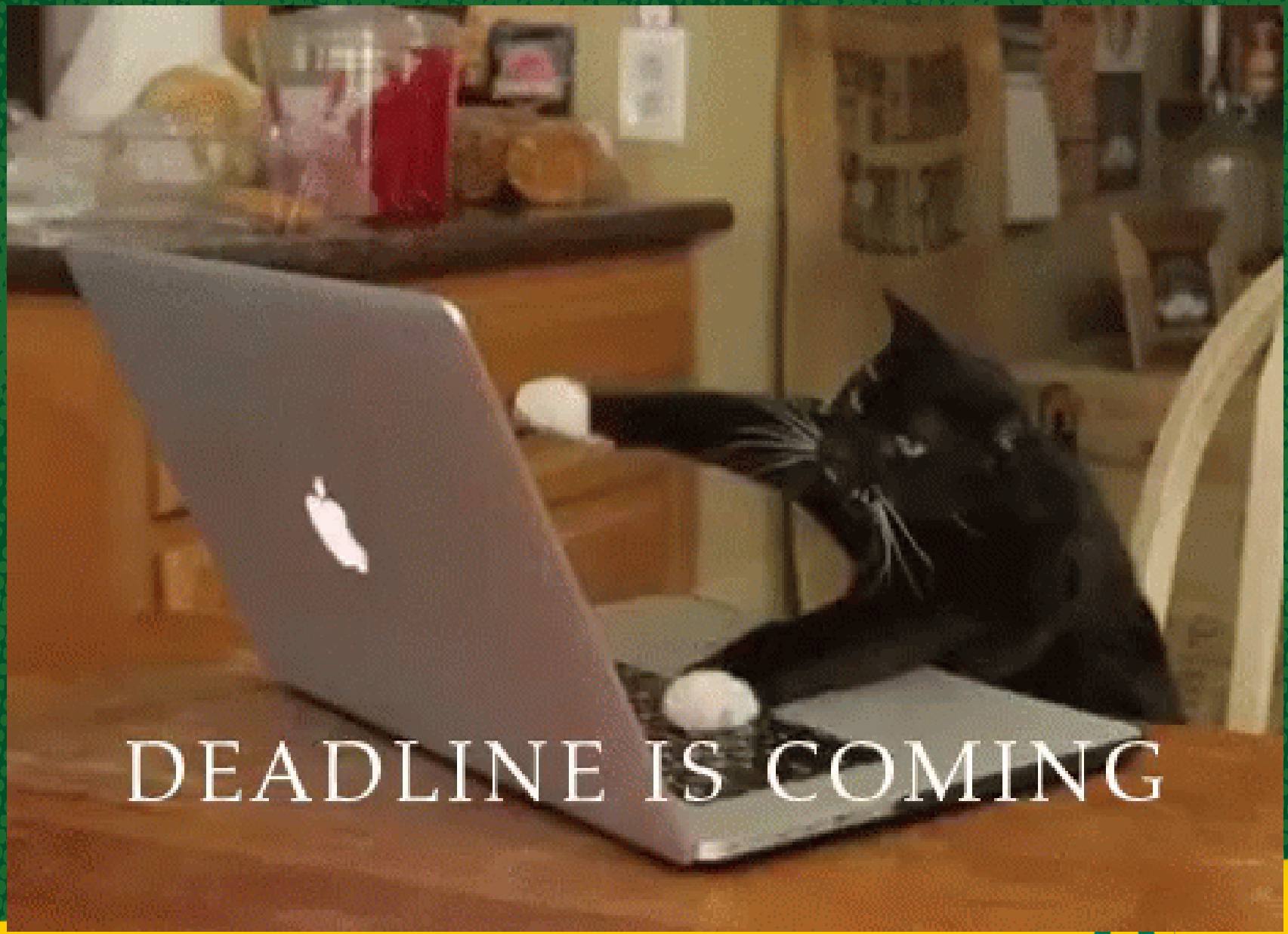
- Need to collect RAM values (used BizHawk)
- Convert Hex to decima and add rambase number and integrate into scenario and data files
- Define actions and rewards for the game.
- Not all RAM values ar so easy to get nor define. (Lua scripts)
- Thanks SethBling


```
 {
: {
erence": "coins",
: "eq"


 {
{
ard": 1.0


 ["DOWN"]],
], ["RIGHT"]],
["B"], ["Y"], ["START"], ["R"], ["R", "A"
 "Y"], ["R", "A", "B"], ["R", "A", "Y"],
"]]
```

So Let's Make Something Work



So Let's Make Something Work

Goals

- Get a program to work nicely with retro
-