# Simulating Different Rules For a Four-Way Intersection

Hyun Min Kim

Mentor: Sylvia Haas

## Project Description

## Intersection Set Up

## Implementing Intersection Rules

## Experimenting with continuous traffic flow

### Creating a function for a continuous flow of cars

### Incorporating Rule #1

I overloaded the Rule #1 function to have “interval” as one of the parameters. This is the interval for cars being put on the road.
To add cars, I had to use the functions previously created to put cars on the roads.

```
In[ ]:= roadRule[listX_, listY_, carsBehindStop_, interval_] := (
          cloneX = listX;
          cloneY = listY;
          iterationCount++;
          carsAtStopSign = {};
          carsInIntersection = {};
          checkStop[listX];
          checkStop[listY];
          checkIntersection[listX];
          checkIntersection[listY];
          If[! IntegerQ[cloneX[[-1]]], carCount++];
          If[! IntegerQ[cloneY[[-1]]], carCount++];
          If[! IntegerQ[cloneX[[carsBehindStop + 1]]],
           changeLaterX = True, changeLaterX = False];
          cloneX = Join[Take[cloneX, carsBehindStop + 1],
             Take[cloneX, {carsBehindStop + 1, Length[cloneX] - 1}]] ;
          MapIndexed[If[First[#2] > carsBehindStop && ! NumberQ[cloneX[[First[#2]]]],
              cloneX[[First[#2], 1, 1, 1]] ++] &, cloneX];
          If[! IntegerQ[cloneY[[carsBehindStop + 1]]], changeLaterY = True,
           changeLaterY = False];
```

```
  cloneY = Join[Take[cloneY, carsBehindStop + 1],
    Take[cloneY, {carsBehindStop + 1, Length[cloneY] - 1}]] ;
  MapIndexed[If[First[#2] > carsBehindStop && ! NumberQ[cloneY[[First[#2]]]],
     cloneY[[First[#2], 1, 1, 2]] ++] &, cloneY];
  If[Length[carsInIntersection] == 0, If[Length[carsAtStopSign] ≠ 0,
    If[Length[carsAtStopSign ] == 1, If[carsAtStopSign[[1, 1, 1, 2]] == 0,
      cloneX[[carsBehindStop + 1]] = listX[[carsBehindStop]];
      cloneX[[carsBehindStop]] = 0;
      cloneX[[carsBehindStop + 1, 1, 1, 1]] ++;,
      cloneY[[carsBehindStop + 1]] = listY[[carsBehindStop]];
      cloneY[[carsBehindStop + 1, 1, 1, 2]] ++;
      cloneY[[carsBehindStop]] = 0;],
     If[cloneX[[carsBehindStop + 1]] = listX[[carsBehindStop]];
       cloneX[[carsBehindStop]] = 0;
       cloneX[[carsBehindStop + 1, 1, 1, 1]] ++;,
       cloneY[[carsBehindStop + 1]] = listY[[carsBehindStop]];
       cloneY[[carsBehindStop + 1, 1, 1, 2]] ++;
       cloneY[[carsBehindStop]] = 0;];]]
  ];
  For[i = carsBehindStop, i > 1, i--,
   If[NumberQ[listX[[i]]] && ! NumberQ[listX[[i - 1]]], {cloneX[[i]] = listX[[i - 1]];
     cloneX[[i, 1, 1, 1]] ++;
     cloneX[[i - 1]] = 0}, cloneX[[i - 1]] = listX[[i - 1]]];
   If[NumberQ[listY[[i]]] && ! NumberQ[listY[[i - 1]]], {cloneY[[i]] = listY[[i - 1]];
     cloneY[[i, 1, 1, 2]] ++;
     cloneY[[i - 1]] = 0}, cloneY[[i - 1]] = listY[[i - 1]]];
  ];
  If[changeLaterX, cloneX[[carsBehindStop + 1]] = 0];
  If[changeLaterY, cloneY[[carsBehindStop + 1]] = 0];
  pushCarX[cloneX, interval];
  pushCarY[cloneX, interval];
  Return[{cloneX, cloneY}]
 )
```

I needed to overload the iteration rule also because I had to implement the interval.

```
iterateRule2[listX_, listY_, cbs_, times_, interval_] := (
  iterationCount = 0;
  carCount = 0;
  NestList[roadRule[#[[1]], #[[2]], cbs] &, {listX, listY}, times])
```

## Incorporating Rule #2

## Running Rule #2 with a continuous flow of cars

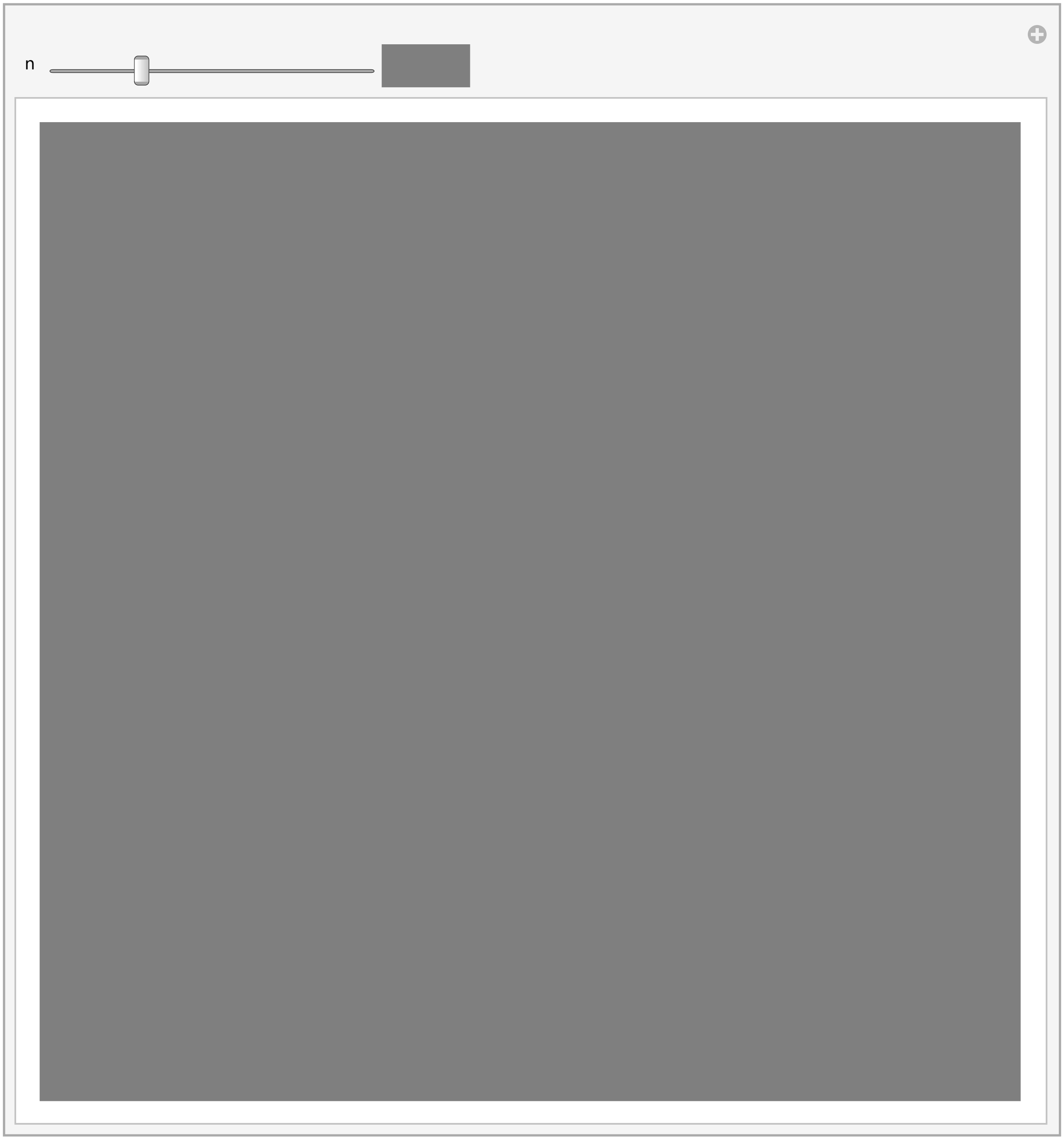
I saved 400 iterations of Rule #2 created in intervals of 2.

```
In[∘]:= iterInter2 = iterateRuleEdit2[carListX, carListY, 10, 4000, 2];
```

I plotted the 400 iterations using manipulate.

```
In[∘]:= Manipulate[
         Graphics[{PointSize[0.025], Select[(Flatten /@ iterInter2)[[n]], ! IntegerQ[#] &],
           hLines[[1]], vLines[[1]]}], {n, 1, 4000, 1}, SaveDefinitions → True]
```

Out[∘]=



## Analyzing Efficiency of Rule #2

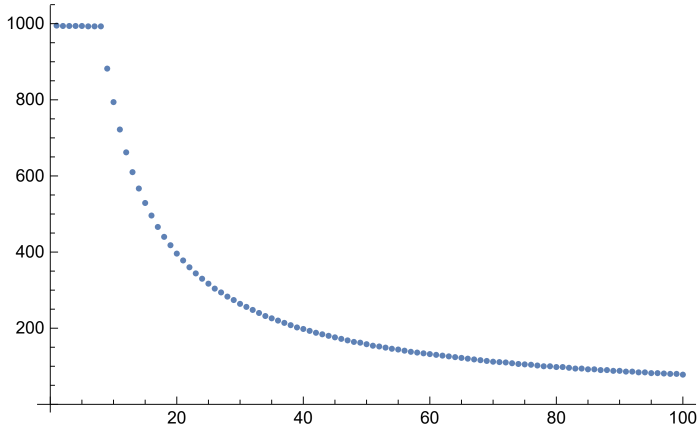
### Defining Variable and Storing Data

## Plotting Data

Plot of data collected by intervals:

```
In[]:= ListPlot[iterInter2List]
```

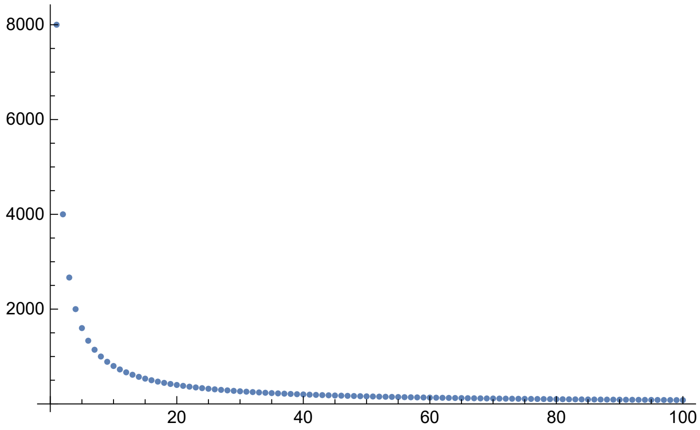
Out[]=



Plot of theoretical number of cars passed without an intersection:

```
In[]:= ListPlot[4000 / Range[100] * 2, PlotRange → Full]
```

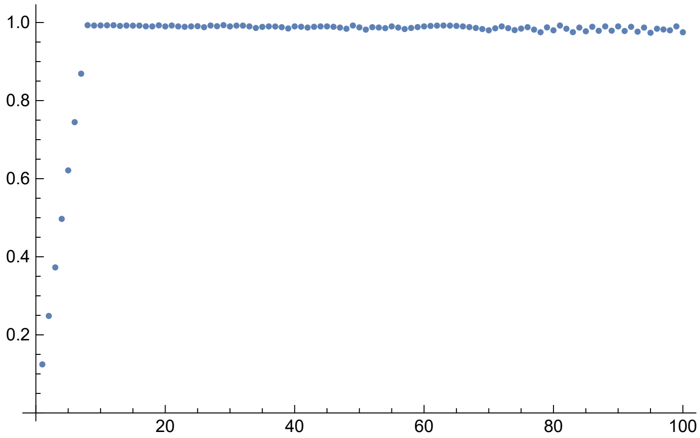
Out[]=



The efficiency graph I plotted after dividing actual/theoretical:

```
In[◦]:= ListPlot[MapThread[Divide, {iterInter2List, 4000 / Range[100] * 2}], PlotRange → Full]
```

Out[◦]=



## Conclusion

There are many steps that could be taken to further this project. In the future, I want to experiment with more types of intersections. Since I only concentrated on four-way intersections, I think it would be very interesting to look into intersections such as roundabouts and various n way intersections. Also, although I am aware that it would be very difficult, I want to figure out how the traffic would be with acceleration.