February 17, 2016

**Traffic simulation** or the simulation of transportation systems is the [mathematical modeling](https://en.wikipedia.org/wiki/Computer_simulation) of transportation systems (e.g., freeway junctions, arterial routes, roundabouts, downtown grid systems, etc.) through the application of computer software to better help plan, design and operate transportation systems. Simulation of transportation systems started over forty years ago,[]](https://en.wikipedia.org/wiki/Traffic_simulation#cite_note-2) and is an important area of discipline in [Traffic Engineering](https://en.wikipedia.org/wiki/Traffic_engineering_(transportation)) and [Transportation Planning](https://en.wikipedia.org/wiki/Transportation_Planning) today. Various national and local transportation agencies, academic institutions and consulting firms use simulation to aid in their management of transportation networks.

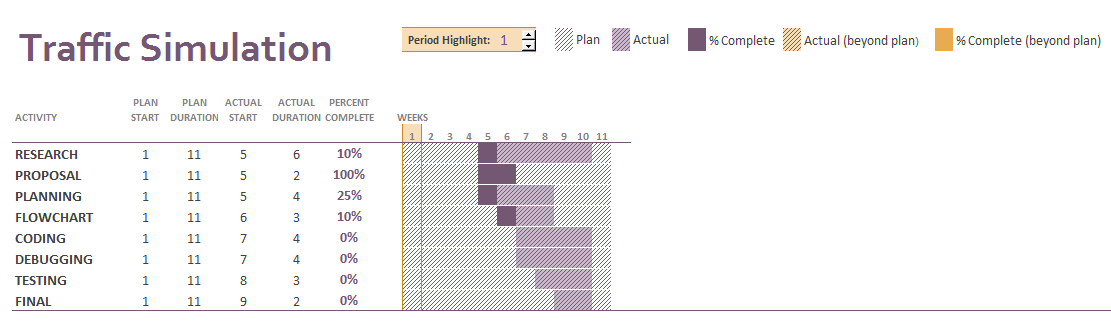
Simulation in transportation is important because it can study models too complicated for analytical or numerical treatment, can be used for experimental studies, can study detailed relations that might be lost in analytical or numerical treatment and can produce attractive visual demonstrations of present and future scenarios.

To understand simulation, it is important to understand the concept of [system state](https://en.wikipedia.org/wiki/Classical_mechanics), which is a set of variables that contains enough information to describe the evolution of the system over time. System state can be either [discrete](https://en.wikipedia.org/wiki/Discrete_time) or [continuous](https://en.wikipedia.org/wiki/Continuous_simulation). Traffic simulation models are classified according to discrete and continuous time, state, and space. Traffic simulation models are useful from a microscopic, macroscopic and sometimes mesoscopic perspectives. Simulation can be applied to both transportation planning and to transportation design and operations. In transportation planning the simulation models evaluate the impacts of regional urban development patterns on the performance of the transportation infrastructure. [Regional planning organizations](https://en.wikipedia.org/wiki/Metropolitan_planning_organization) use these models to evaluate what-if scenarios in the region, such as air quality to help develop [land use](https://en.wikipedia.org/wiki/Land_use) policies that lead to more [sustainable travel](https://en.wikipedia.org/wiki/Sustainable_transport). On the other hand, modeling of transportation system operations and design focus on a smaller scale, such as a highway corridor and pinch-points. Lane types, signal timing and other traffic related questions are investigated to improve local system effectiveness and efficiency. While certain simulation models are specialized to model either operations or system planning, certain models have the capability to model both to some degree.

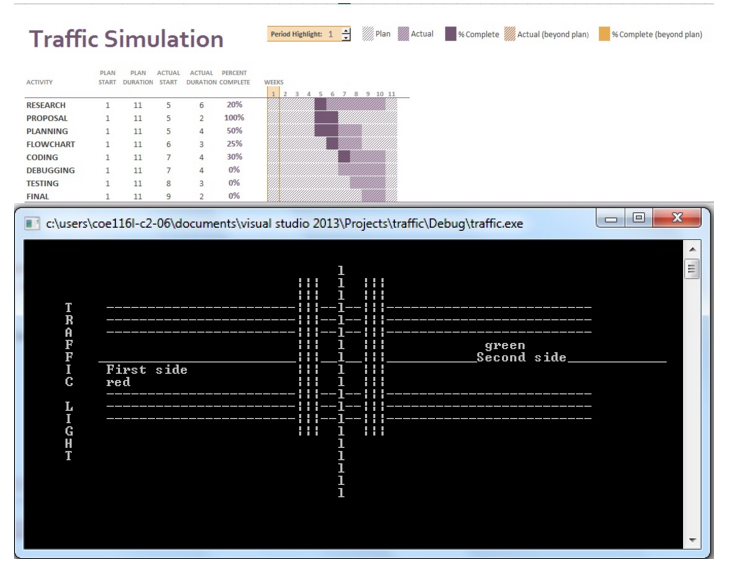
Whether it is for planning or for systems operations, simulations can be used for a variety of [transportation modes](https://en.wikipedia.org/wiki/Mode_of_transport).

 February 25, 2016

Our group is planning to apply some concepts from linear algebra in creating an object oriented program that will simulate a traffic.



 March 3, 2016



March 10, 2016

This program simulates traffic lights and we are planning to add some applications of Queues for counting the number of cars.

traffic.h

#include<iostream>

using namespace std;

class traffic

{

private:

struct Node

{

int info;

Node \* next;

};

Node \* front = NULL;

Node \* front1 = NULL;

Node \* rear = NULL;

Node \*rear1 = NULL;

public:

void enqueue(int x, int y)

{

struct Node \* temp = new Node;

if (y == 0)

front = NULL;

temp->info = x;

temp->next = NULL;

if (front == NULL)

{

front = temp;

}

else

{

rear->next = temp;

}

rear = temp;

}

void enqueue1(int x, int y)

{

struct Node \* temp = new Node;

if (y == 0)

front1 = NULL;

temp->info = x;

temp->next = NULL;

if (front1 == NULL)

{

front1 = temp;

}

else

{

rear1->next = temp;

}

rear1 = temp;

}

int dequeue(int x, int y)

{

struct Node \* temp = front;

if (front == NULL)

return y;

else if (x == 0)

{

while (temp != NULL)

{

y = temp->info + y;

temp = temp->next;

}

y--;

return y;

}

else

{

if (y != 0)

y--;

return y;

}

}

int dequeue1(int x, int y)

{

struct Node \* temp = front1;

if (front1 == NULL)

{

return y;

}

else if (x == 0)

{

while (temp != NULL)

{

y = temp->info + y;

temp = temp->next;

}

y--;

return y;

}

else

{

if (y != 0)

y--;

return y;

}

}

int display()

{

int x = 0;

struct Node \*temp = new Node;

temp = front;

if (front == NULL)

return x;

else

{

while (temp != NULL)

{

x = temp->info + x;

temp = temp->next;

}

return x;

}

}

int display1()

{

int x = 0;

struct Node \*temp = new Node;

temp = front1;

if (front1 == NULL)

return x;

else

{

while (temp != NULL)

{

x = temp->info + x;

temp = temp->next;

}

return x;

}

}

};

traffic.cpp

// traffic4.cpp : Defines the entry point for the console application.

//

/\*#include "stdafx.h"

#include <iostream>

#include <stdio.h>

#include <time.h>

using namespace std;

void fiftyseconds()

{

char choice = 'N';

int x = 0, y = 0;

time\_t start;

time\_t current;

time(&start);

do {

time(&current);

} while (difftime(current, start) < 10.0);

}

void twoseconds()

{

time\_t start;

time\_t current;

time(&start);

do { time(&current);

} while (difftime(current, start) < 2.0);

}

void redlight()

{

cout<<"Lane1 - RED; Lane2 - RED;";

printf("\tLane3. - GREEN; Lane4 - GREEN\n");

fiftyseconds();

printf("Lane1 - RED; Lane2 - RED; \tLane3. - YELLOW; Lane4 - YELLOW\n");

twoseconds();

printf("Lane1 - RED; Lane2 - RED; \tLane3. - RED; Lane4 - RED\n");

twoseconds();

printf("Lane1 - GREEN; Lane2 - GREEN; \tLane3. - RED; Lane4 - RED\n");

fiftyseconds();

printf("Lane1 - YELLOW; Lane2 - YELLOW; \tLane3. - RED; Lane4 - RED\n");

twoseconds();

printf("Lane1 - RED; Lane2 - RED; \tLane3. - RED; Lane4 - RED\n");

twoseconds();

}

int main(){

printf("Stoplight 1.0\n\n");

while (1)

redlight();

return 0;

}\*/

#include "stdafx.h"

#include "traffic4.h"

# include <iostream>

#include<Windows.h>

using namespace std;

void gotoxy(int, int);

int main()

{

traffic k;

system("cls");

char cg[14] = "TRAFFIC LIGHT";

char car[20] = "Car Interval: 2 sec";

char lane[14] = "One way Lanes";

char cont[34] = "Would you like to continue?(Y/N):";

char choice = 'Y';

for (int i = 0; i<14; i++)

{

gotoxy(5, 5 + i);

cout << cg[i];

Sleep(100);

}

for (int i = 0; i<20; i++)

{

gotoxy(5 + i, 0);

cout << car[i];

Sleep(100);

}

for (int i = 0; i<14; i++)

{

gotoxy(5 + i, 1);

cout << lane[i];

Sleep(100);

}

char c[14] = "|||||||||||||";

char d[60] = "-----------------------------------------------------------";

char e[70] = "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

char f[20] = "lllllllllllllllllll";

gotoxy(10, 8);

cout << "First Lane";

gotoxy(41, 17);

cout << "Second Lane";

gotoxy(10, 12);

cout << d;

gotoxy(10, 13);

cout << d;

gotoxy(10, 14);

cout << d;

gotoxy(10, 5);

cout << d;

gotoxy(10, 6);

cout << d;

gotoxy(10, 7);

cout << d;

for (int i = 0; i<70; i++)

{

gotoxy(9 + i, 9);

cout << e[i];

Sleep(100);

}

for (int i = 0; i<14; i++)

{

gotoxy(33, 3 + i);

cout << c[i];

}

for (int i = 0; i<14; i++)

{

gotoxy(34, 3 + i);

cout << c[i];

}

for (int i = 0; i<14; i++)

{

gotoxy(35, 3 + i);

cout << c[i];

}

for (int i = 0; i<14; i++)

{

gotoxy(41, 3 + i);

cout << c[i];

}

for (int i = 0; i<14; i++)

{

gotoxy(42, 3 + i);

cout << c[i];

}

for (int i = 0; i<14; i++)

{

gotoxy(43, 3 + i);

cout << c[i];

}

for (int i = 0; i<20; i++)

{

gotoxy(38, 2 + i);

cout << f[i];

Sleep(100);

}

int sig; sig = 41;

int sig1 = 18;

int z = 0;

int ctr = 0;

do

{

int x = 0;

int y = 0;

if (x == 0)

{

gotoxy(41, 18);

cout << "red";

}

if (sig == 10)

{

sig = 41;

sig1 = 18;

}

else

{

sig = 10;

sig1 = 9;

}

gotoxy(sig, sig1);

cout << "red ";

Sleep(1500);

gotoxy(sig, sig1);

cout << "green ";

for (int i = 0; i < 5; i++)

{

if (ctr % 2 != 0)

{

k.enqueue(1, x);

z = k.display();

gotoxy(10, 10);

cout << "Cars on First Lane: " << z;

y = k.dequeue1(x, y);

gotoxy(41, 19);

cout << "Cars on Second Lane: " << y;

Sleep(2000);

x++;

}

else

{

k.enqueue1(1, x);

z = k.display1();

gotoxy(41, 19);

cout << "Cars on Second Lane: " << z;

y = k.dequeue(x, y);

gotoxy(10, 10);

cout << "Cars on First Lane: " << y;

Sleep(2000);

x++;

}

}

gotoxy(sig, sig1);

cout << "yellow ";

if (ctr % 2 != 0)

{

k.enqueue(1, x);

z = k.display();

gotoxy(10, 10);

cout << "Cars on First Lane: " << z;

y = k.dequeue1(x, y);

gotoxy(41, 19);

cout << "Cars on Second Lane: " << y;

Sleep(2000);

x++;

}

else

{

k.enqueue1(1, x);

z = k.display1();

gotoxy(41, 19);

cout << "Cars on Second Lane: " << z;

y = k.dequeue(x, y);

gotoxy(10, 10);

cout << "Cars on First Lane: " << y;

Sleep(2000);

x++;

}

gotoxy(sig, sig1);

cout << "red ";

Sleep(1500);

if (ctr % 2 != 0)

{

k.enqueue(1, x);

z = k.display();

gotoxy(10, 10);

cout << "Cars on First Lane: " << z;

Sleep(2000);

x++;

}

else

{

k.enqueue1(1, x);

z = k.display1();

gotoxy(41, 19);

cout << "Cars on Second Lane: " << z;

Sleep(2000);

x++;

}

if (ctr % 2 != 0)

{

for (int i = 0; i<34; i++)

{

gotoxy(5 + i, 2);

cout << cont[i];

Sleep(100);

}

cin >> choice;

if (choice != 'Y')

return 0;

else

{

for (int i = 0; i < 35; i++)

{

gotoxy(5 + i, 2);

cout << " ";

}

}

}

ctr++;

} while (1>0);

return 0;

}

void gotoxy(int x, int y)

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

\_COORD pos; pos.X = x; pos.Y = y;

SetConsoleCursorPosition(hConsole, pos);

}

JOURNAL