T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

LINE – STATION - PASSENGER SIMULATION

ALPER YAŞAR

SUPERVISOR
ASSOC. PROF. MEHMET GÖKTÜRK

GEBZE
2022

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

LINE – STATION - PASSENGER SIMULATION

ALPER YAŞAR

SUPERVISOR
ASSOC. PROF. MEHMET GÖKTÜRK

2022
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 31/08/2021 by the following jury.

JURY

Member (Supervisor)	:	Assoc. Prof. MEHMET GÖKTÜRK
Member	:	Prof. İbrahim Soğukpınar

ABSTRACT

This project is a public transport bus service simulation. Also name as ’Urban Bus Simulation’. In this project we are holding lines information data. This data are schedule time and stations name. Schedule time is the time when the bus departure time. When departure time is come a bus is creating and running on line. Each minutes creating random passengers. Each passengers have current station name and destination name. Stations holding passengers. When the correct bus for stations the passenger get in to bus.

The main purpose of this when 2 buses approached the station passengers what will do? Which bus is the passengers choose?

Keywords: bus, line, station, urban, passengers, simulation.

CONTENTS

LIST OF FIGURES

1. SIMULATION

A simulation is the imitation of the operation of a real-world process or system over time.[1] Simulations require the use of models; the model represents the key characteristics or behaviors of the selected system or process, whereas the simulation represents the evolution of the model over time. Often, computers are used to execute the simulation. There are many game engines for 2D simulation game.

1.1. Godot Engine

Godot is open source and completely free. It's been praised for being very lightweight and fast, and it's capable of creating both 3D and 2D games. It functions a little differently than other engines, and it has its own programming language: GDScript. GDScript is similar to Python, so it shouldn't be a challenge to learn for those with a Python background, or for beginners. It also allows for visual scripting with connectable blocks and scripting in other languages, like C++ and C#. 1.1

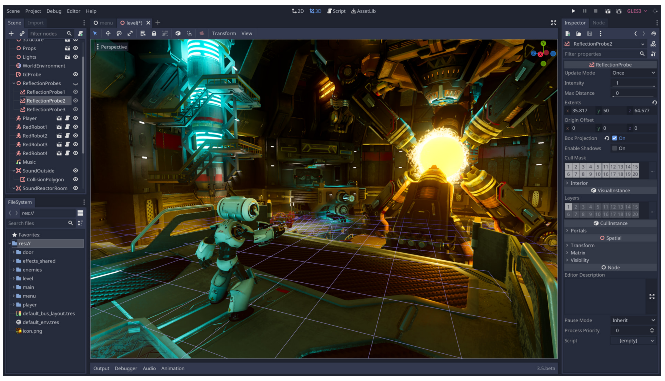

Figure 1.1: Godot is open-source

1.2. Unity

Unity is one of the most used game engines of all time, and many people are already familiar with it. The engine is built around making 3D games, but Unity is also good for 2D as well. It has a massive community full of helpful tutorials, and it also has the Unity Asset Store, which is full of useful tools and assets to help game development.

Unity has been used to produce a few well known 2D games, like Ori and the Blind Forest, Cuphead, and West of Loathing. Unity is free to download and use for projects with funding or revenue under $100k. This makes it affordable for indie developers, although it moves up to a more expensive pricing model after that cap is reached. 1.2.

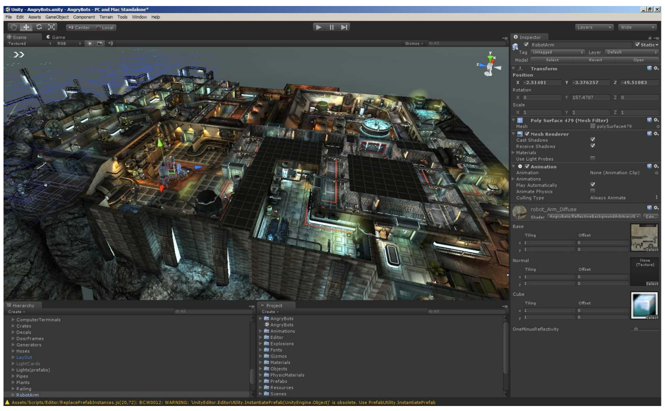
Figure 1.2: Unity is very popular

1.3. GameMaker Studio

GameMaker Studio excels at starting and making games quickly, with an extremely rapid setup time and an easy to learn programming language, it's easy to get your ideas into code right away. It also features its own visual programming language called Drag and Drop, making this a good 2d game engine for no coding. The desktop license starts at $99 1.3.

Figure 1.3: GameMaker Studio

1.4. Other Engines

There are more engines for using :

- Corona
- RPG Maker
- libGDX
- Ren'py

1.5. Why did I used Unity

The animation is considered one of the most important components of any 2D game, and Unity 2D makes the 2D animation much simpler. It comes with a simpler interface and controls, which you can use to rig sprites and set up bones to create a smooth animation. Also, it allows you to import the PSD to Unity, and you can access the layered artwork directly from the software. Also unity allow to scripting to user.

1.5.1. What is scripting in Unity?

Scripting tells our GameObjects how to behave; it's the scripts and components attached to the GameObjects, and how they interact with each other, that creates your gameplay. Now, scripting in Unity is different from pure programming. Unity runs in a big loop. It reads all of the data that's in a game scene.

1.5.2. What languages can you use in Unity?

A script must be attached to a GameObject in the scene in order to be called by Unity. Scripts are written in a special language that Unity can understand. And, it's through this language that we can talk to the engine and give it our instructions.

The language that's used in Unity is called C#. All the languages that Unity operates with are object-oriented scripting languages. Like any language, scripting languages have syntax, or parts of speech, and the primary parts are called variables, functions, and classes.

2. CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

2.1. Purpose of Class Diagrams

1. Shows static structure of classifiers in a system.
2. Diagram provides a basic notation for other structure diagrams prescribed by UML.
3. Helpful for developers and other team members too.
4. Business Analysts can use class diagrams to model systems from a business perspective.

2.2. What is a Class?

A description of a group of objects all with similar roles in the system, which consists of:

2.2.1. Structural features

(attributes) define what objects of the class "know".

- Represent the state of an object of the class.
- Are descriptions of the structural or static features of a class.

2.2.2. Behavioral features

(operations) define what objects of the class "can do".

- Define the way in which objects may interact.
- Operations are descriptions of behavioral or dynamic features of a class.

2.2.3. The class diagram of LINE – STATION - PASSENGER SIMULATION :

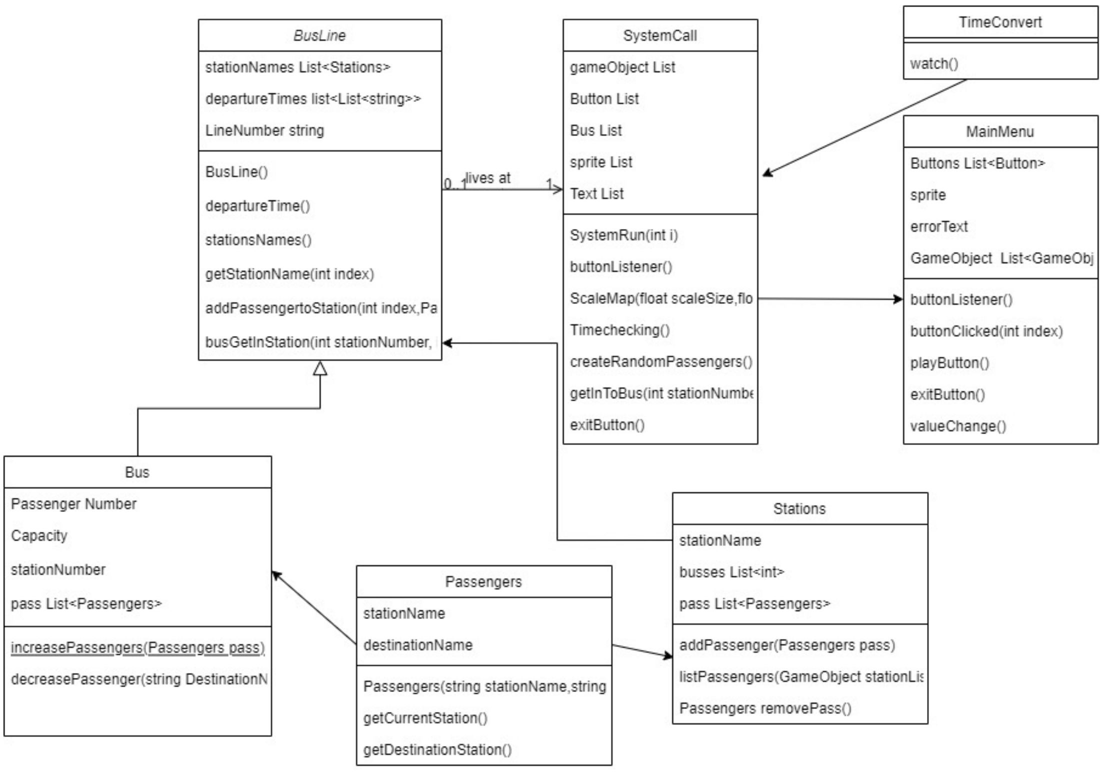

Figure 2.1: Class diagram of project

3. PROJECT

This project is a public transport bus service simulation. In this simulation the buses has departure time [2]. Each bus line have own stations [2]. Each bus has capacity. The average of each distance is 2 minutes and buses changing its position every 24 seconds. When the bus arrive to station, if there is anybody at station for bus, it will wait until the passenger get in to bus.

Each passengers get in to bus time is 3 seconds. Each minutes creating passengers randomly. Each passenger waiting his/her current station for bus and has destination station. When passenger come to its destination station he/she is going to get off from bus.

The main purpose of this, when 2 bus arrive to 1 station the passenger choosing a bus. If the bus which front of has more capacity than behind bus passenger plus passengers at station, the front bus is not waiting for passengers and all passenger getting in to behind bus. Otherwise the passengers, in front of the queue, get in to front bus. When the each passenger get in to the bus this condition checking.

3.1. Main Menu

In this part the user choose how simulation will be :

- Line name.

  The green button means it is ready for using. When choose the green button is going to be highlighted.

  The grey button means, this line is not ready for usable.

  When choose grey and push to play button, the simulation will be run. if the user choose the grey one there will be error for choose one from green buttons.

- Value of passengers.

  If user did not enter any the simulation will run. Assign initial value to them. In the button the transparents values are the initial value of each button.

  Sitting Passengers: The chair capacity of bus.

  Standing Passengers: The capacity of stanting passengers.

Random Passengers Number: The number of passengers whose creating each minutes.

Acceleration: System runnig as real time. We can accelerate the system. If acceleration number is bigger than 1, the system is going to be accelerated. This means that divide each time to acceleration value and each steps going to be that time.

Play: System run button.

Exit: Quit from program.

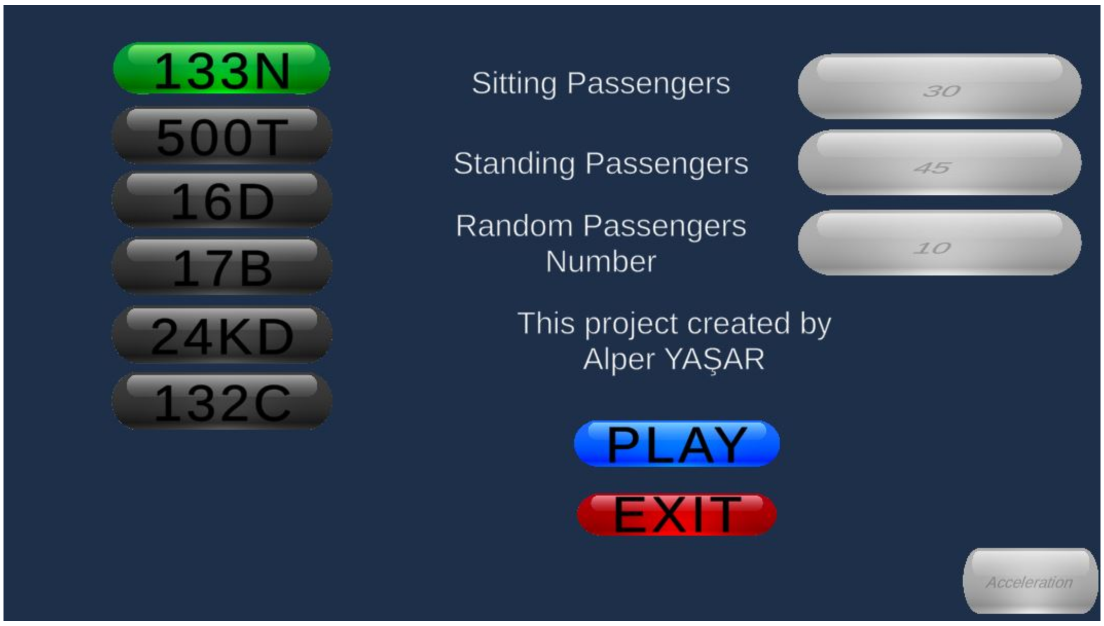

Figure 3.1: Main Menu

3.2. SIMULATION RUN

When push the play button system will be run. First of all the system starting checking the time. This function calling on update function. Because we need check time. Multiply time with acceleration value and assing to new variable. Check the time if it's a multiple of sixty. If the mode of the time is equal to 0, change the minutes. Also random passengers creating each minutes. This function calling in Timechecking() function. This function returning boolean value. In addition in this function we are checking schedule time. When bus departure time has came this function returning true. Otherwise returning false.

Algorithm 1 Timechecking()

```
$virtualTime \leftarrow time \times acceleration$
**if** $virtualTime \mod 60 == 0$ **then**
    $XtempWatch \leftarrow timeConvert.watch()$
    print $tempWatch$
    $createRandomPassengers()$
    **if** $buses.timeCheck(tempWatch)$ **then**
        return $true$
    **end if**
**end if**
return $false$
```

3.2.1. Data

We need to hold schedule time and station names. Schedule time is that time of each bus departure of line. Each line has own schedule time. When the departure time buses are creating. Station names are the station on the line. These data are holding in different excel table. When the system start they are reading from excel to list on class.

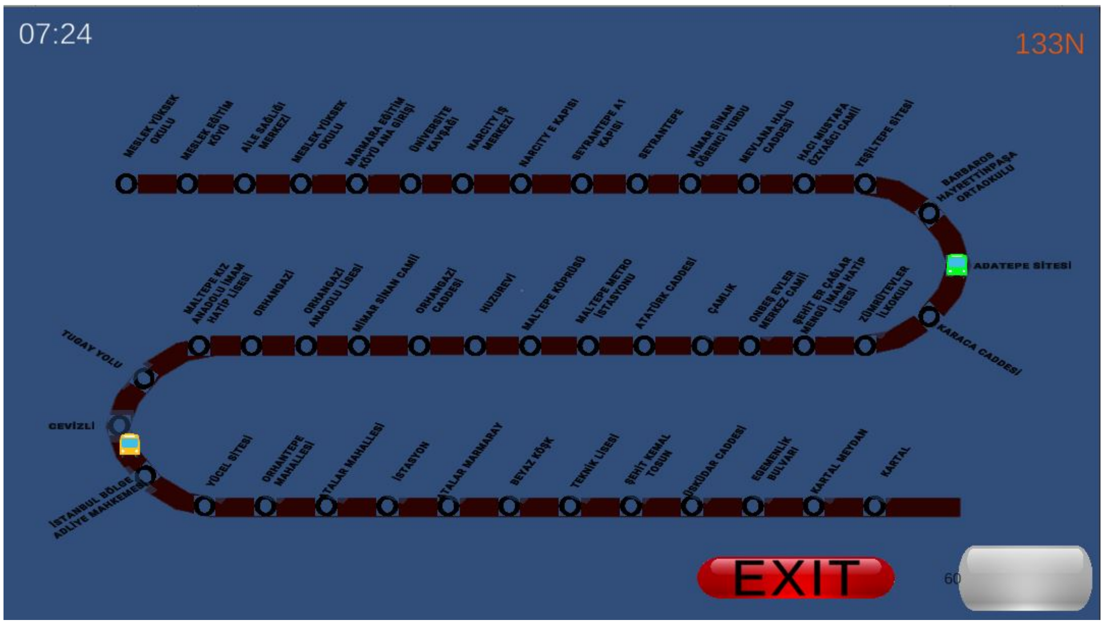

Figure 3.2: Simulation is running

3.2.2. Create Random Passenger

If the user enter any value of enterance the simulation this is goning to be value of passengers whose creating each minutes. Otherwise the value assigned 10 as initial value. Each minutes creating minimum 1 maximum 10 passengers (or user specified value).

Algorithm 2 createRandomPassengers()

```
  $randomNumberofPass \leftarrow Random0toMaxPassengersValue$
  for $i0torandomNumberofPass$ do
      $XfirstStation \leftarrow Random0tobuses.getlineLenght() - 1$
      $XsecondStation \leftarrow RandomfirstStationtobuses.getlineLenght() - 1$
      $pass \leftarrow newPassengers(firstStation, secondStation)$
      if $buses.addPassengertoStation(firstStation, pass);$ then
      end if
  end for
```

Firstly calculation how many passengers will create on this minute (0 to 10(or user specified value)). After this calculation in loop creating each passengers. Each passengers has current station and destination station. First station value is 0 to bus line lenght. After this assignment second station value is first station value to bus line lenght. When find the stations of passenger, it will be create and adding to first to station as a passenger.

3.2.3. SystemRun

After checking minutes if Timechecking() returned true that's mean this is the time for a bus departure time. And uptade function calling SystemRun(index) function. İndex initially assinged 0, because this is buses value. Bus holding in list.

In scene there is a bus with position and image. When SystemRun() function is called, firstly creating new bus like as bus which in scene.

$tempObject \leftarrow Instantiate(bus, bus.transform.position, bus.transform.rotation)$

Changing the parent. Because We are adding to busTable. busTable is the line panel. Purpose of this changing when click the any station the busTable size is changing. And the position of new object on the line is saving. Set position of new bus as first station for begining. Adding the new bus the buses list.

$tempObject.transform.SetParent(busTable.transform)$

$tempObject.transform.position \leftarrow station[0].position$

$clone.Add(tempObject)$

For holding the bus informations, creating and adding new Bus script.

```
$BustempBus = newBus()$
$buss.Add(tempBus)$
```

After assigning all variable bus travelled on line in a loop. We get approximately arrival time of bus to new station is 2 minutes. For this approach bus making progress at 5 steps between 2 stations. So each steps is 24 seconds. After position changed the bus will wait 24 seconds. This wait time changing with acceleration time. If the acceleration is wait time is 12 seconds (24/2). If the bus in a station passengers will get in to bus.

```
while $station < buslinelenght$ do
    if $station$ then
        $setclone[index].position$
        $waituntilgetInToBus(station, index)$
        $wait24seconds$
    else
        $setclone[index].position$
        $wait24seconds$
    end if
end while
$removeclone[index]$
```

All this is function pseudocode is :

Algorithm 3 SystemRun(index)

```
$tempObject \leftarrow Instantiate(bus, bus.transform.position, bus.transform.rotation)$
$tempObject.transform.SetParent(busTable.transform)$
$tempObject.transform.position \leftarrow station[0].position$
$clone.Add(tempObject)$
$BustempBus = newBus()$
$buss.Add(tempBus)$
while $station < buslinelenght$ do
    if $station$ then
        $setclone[index].position$
        $waituntilgetInToBus(station, index)$
        $wait24seconds$
    else
        $setclone[index].position$
        $wait24seconds$
    end if
end while
$removeclone[index]$
```

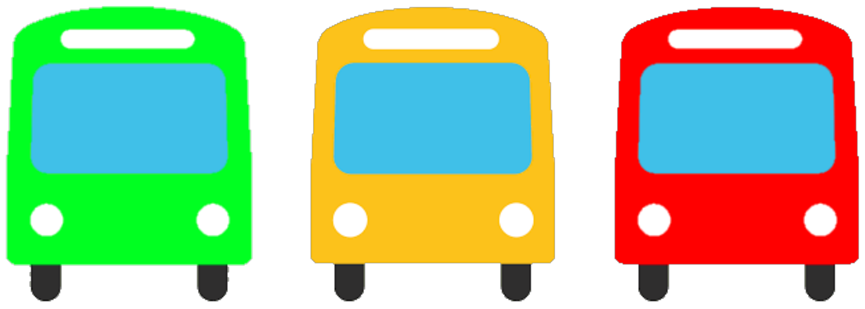
(a) (b) (c)

Figure 3.3: (a) Empty sit(b) Empty Stand (c) full

3.2.4. Get In To Bus

When the bus get arrive the station, firstly checking if anybody get off this station. If there is somebody getting of at this station the bus will wait 3 seconds for everybody. Decreasing the bus passengers number and removing the passengers who get off. We are doing this first, because of update the bus situation.

After that checking the bus color. If there is any empty chair the bus colour is green. If the capacity is full the colour is red. Otherwise colour is yellow 3.3.

If there is somebody in station and just 1 bus add this passenger to bus and check colour again.

Else check how many bus in station. Firstly check the bus capacity. If number of passengers in the bus bigger than other number of passengers in bus and the passengers who are in the station, then bus do not wait in this station. All passengers choose other buses.

Secondly, if difference less than passengers in station, passenger who are in front of queue get in the first arrived bus. Like the real world. In real world the passengers in front of the queue choosing the first arrived bus and other passengers looking the bus capacity and change if necessary. When a passenger get in the bus check the colour and check difference between number of passenger of in front of bus with other number of passengers in bus and the passengers who are in the station. İf difference is bigger than the which in front of the queue will going to move on. Otherwise behind passenger of passenger who has just get in the bus will get same bus. This loop continue until all passengers get in to appropriate bus or full of the bus capacity.

3.3. Passengers at Station

Each station names taking from data. And holding some informations in class. In station class holding station name, passengers list and bus list. Holding bus list purpose calculation and getting the passengers to correct bus in we just talk about previous section.Holding passengers list purpose is show and get them in into bus. We will talk about show passengers at station in this section.

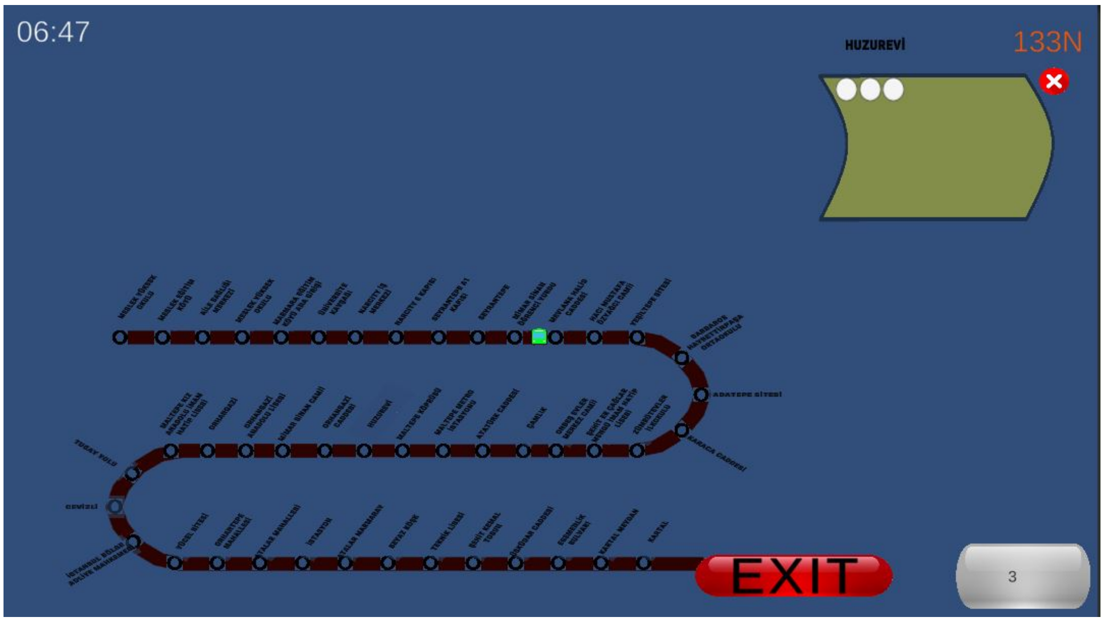

Figure 3.4: Simulation is running

Each station is button. When click the button a window is openning. We are seeing how many passengers at this station. When a bus arrived the station and passengers are going to start getting in to bus each passengers removing from station and the white circle is deleting. When this window opened the bus line resizing. Bus line making smaller. When the station window closed bus line returning same size. We will talk about next section.

3.4. ScaleMap

Bus line size changing when the station wind opening or closing. This scale changing taking half seconds. When the station window opened the size is going to be smaller. So scale is going to adding with minus value.

$$Scale = -0.3 * 0.02985489$$

The table scale adding with scale/5.

$$busTable.localScale + scaleSize/5$$

$$busTable.position - 0.4$$

This dividing scale to 5 because it seems smalling in times. This progress in happining in 5 steps and total seconds is 0.5. Position change is 0.4. For smalling this table substracting and adding for change size back.

```
**for** i 0 to 5 **do**
    $busTable.localScale \leftarrow busTable.localScale + scaleSize/5$
    $busTable.position \leftarrow busTable.position - 0.4$
    wait 0.1 seconds
**end for**
```

3.5. Show Bus Passengers

When we click the bus we can see how many people get in to this bus. In first statement "*sit*" means how many chairs has this bus and how many passengers get in to bus (like "*30/12*").

In second statment "*stand*" means how many people can stand in this bus and how many people are standing at now in this bus (like "*45/4*") 3.5.

sit 30/12
stand 45/4

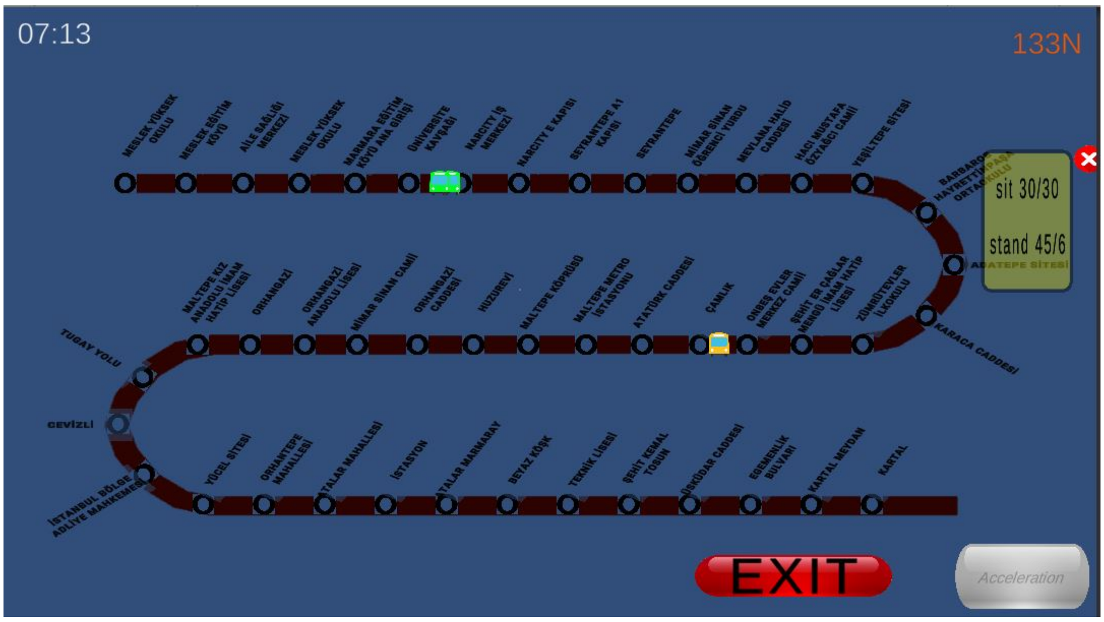

Figure 3.5: Bus window

Also if 2 bus overlap on simulation when we click the bus 2 window is opening.

We can see capacity 2 of that buses. 3.6

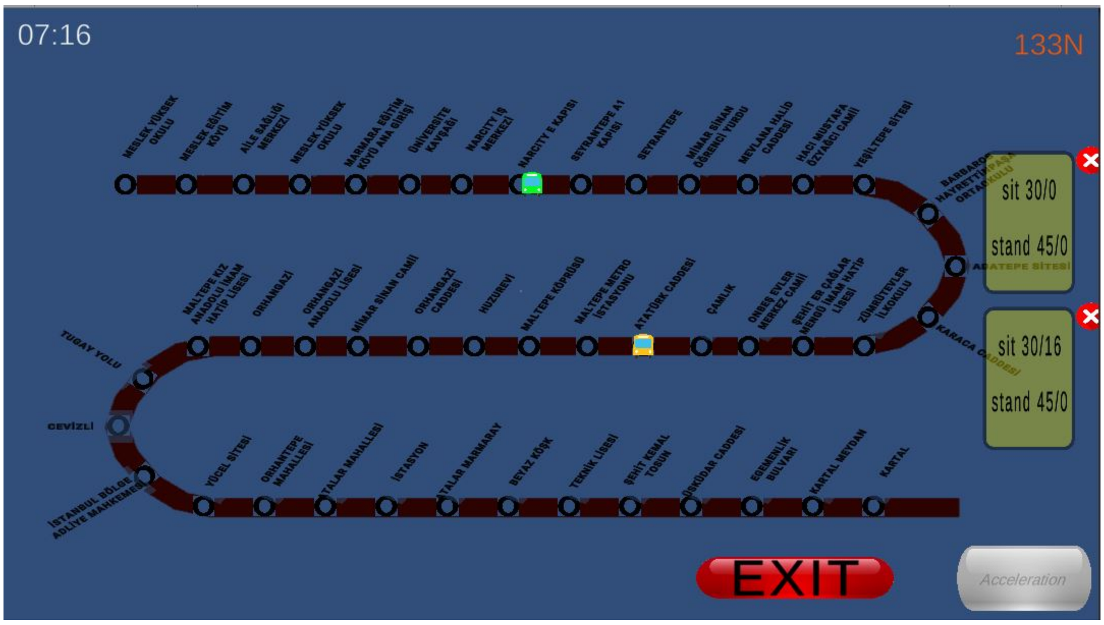

Figure 3.6: Bus window

BIBLIOGRAPHY

[1] J. Banks; J. Carson; B. Nelson; D. Nicol (2001). *Discrete-Event System Simulation*, Prentice Hall. p. 3. ISBN 978-0-13-088702-3.

[2] David Meignan *, Olivier Simonin, Abderrafia^a Koukam *Simulation and evaluation of urban bus-networks using a multiagent approach* , Systems and Transportation Laboratory (S.e.T.), University of Technology Belfort Montbe ́liard, 90000 Belfort, France

[3] Mohamed Tliga, Neïla Bhouri, *A Multi-Agent System for Urban Traffic and Buses Regularity Control.*

[4] Liu, Ronghui and Sinha, Shalini (2007). *Modelling Urban Bus Service and Passenger Reliability.* . In: The Third International Symposium on Transportation Network Reliability, 19-20 July 2007, The Hague, Netherlands.

[5] Tomasz Schelenz a,n , Angel Suescun , MariAnne Karlsson, Li Wikstrom ̈ . *Decision making algorithm for bus passenger simulation during the vehicle design process* .

[6] Erdal Yılmaz. *KENTİÇİ OTOBÜS TAŞIMACILIĞINDA BİR MODEL ÖNERİSİ, SİMÜLASYON TEKNİĞİ İLE PERFORMANS DEĞERLEMESİ* .

[7] https://www.iett.istanbul