FINAL PROJECT
REAL-TIME SYSTEMS

SCUOLA SUPERIORE SANT'ANNA

EMBEDDED COMPUTING SYSTEMS

Trains

Authors:
Lorenzo D'Agostino

Email:
l.dagostino5@studenti.unipi.it

Date: May 7, 2019

1 Abstract

The project simulates a station with 8 platforms and 2 entrances per direction. The trains have an arrival and departure time, read from a file, and are scheduled by the train manager and dispatched to the available platforms, considering priorities of trains.
The layout of the station is the following:

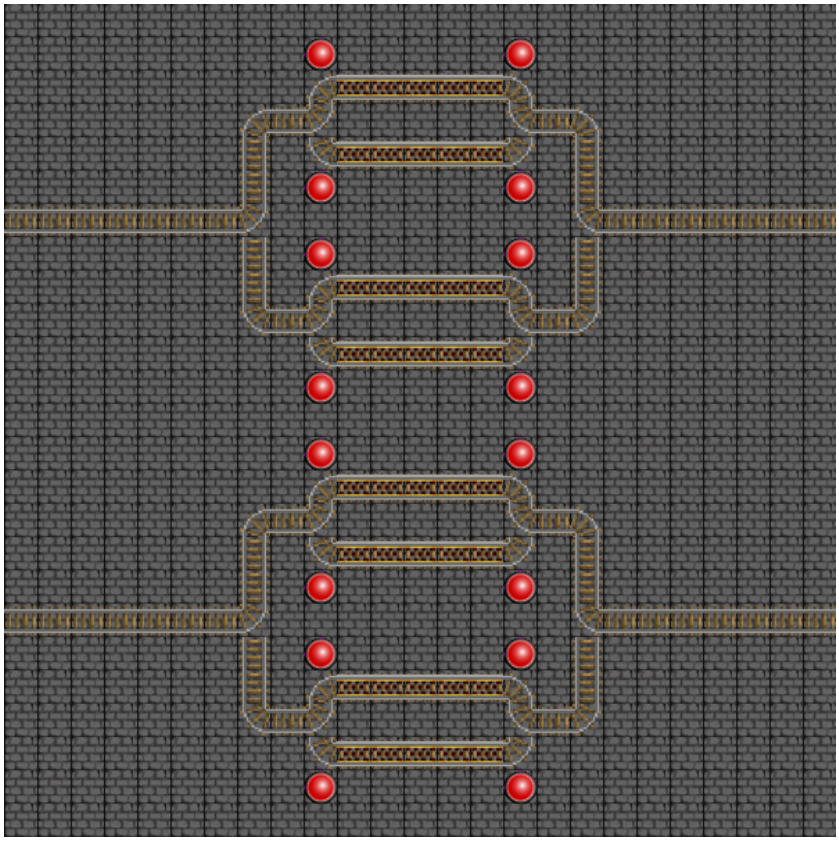

Figure 1: Station

2 Description

2.1 Station

The station is represented by a two-dimensional rectangular box. This box is then divided into a 25x25 matrix. Each slot of the matrix is considered a ”cell” and those cells compose the station.
A cell may contain a track, if so it must contain permitted movement directions.
A track may be a dock track, where train are supposed to stop.

2.2 Train

The station is populated by trains that enter from two possible tracks for each direction.
Every train evolves semi-autonomously: it may remain in a given status doing what the status is meant to do or check/set a global condition and moving to another status.
The status are analyzed in the following part.

ARRIVING
The train has been loaded from the file and it's waiting for it's arrival time. It will move to DOCKING after being given a docking platform

DOCKING
The train is currently moving in the station, following tracks. It will move until reaching the docking point, then it will stop and move into DOCKED status.

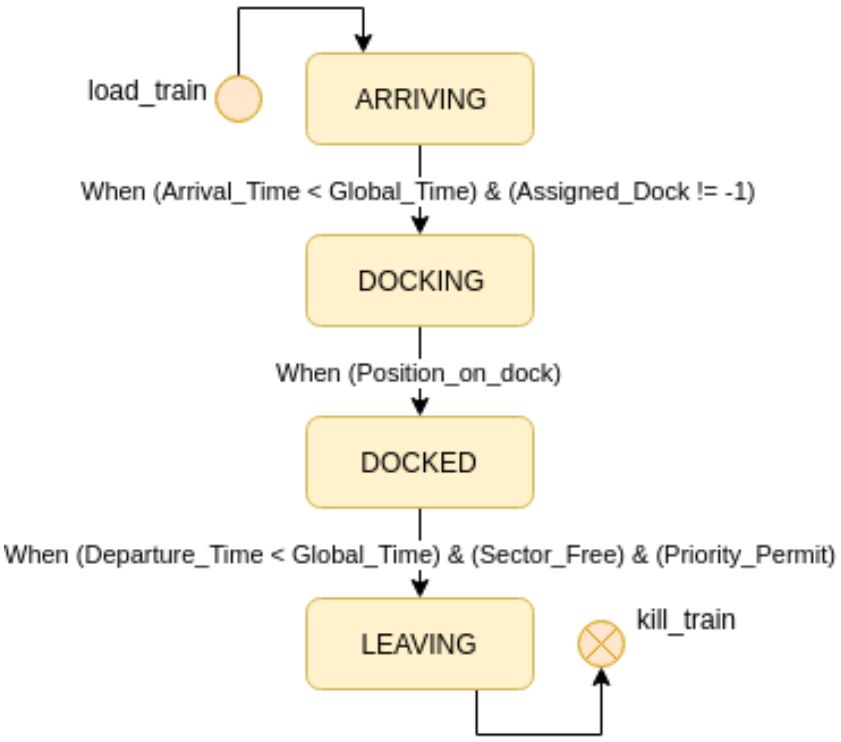

Figure 2: Train Status FSM

DOCKED
The train can be in two different situations when DOCKED. It may be waiting for it's departure time to come, or it might have arrived but the train is waiting to leave because either the track is occupied by an incoming train or by a train with higher priority.
When all requisites are reached the train will move to LEAVING status.

LEAVING
The train is leaving the station, moving on tracks until it reaches the end of the station. Then the task is terminated, the train deallocated and the track freed.

3 Interface and User Interactions

3.1 Interface

The main and only interface part is the ”station”. Every cell in the station has a background called ”tile” (Allegro Bitmap), on which there can be a ”track” (Allegro Sprite) if the cell has two possible directions of movement. Trains (Allegro Sprites) move on the station, not on the cells, and when on a specific cell they can perform the movement permitted by that specific cell.

3.2 User Interactions

During the simulation the user can interact in 2 ways:

- Spawning a maximum priority train
- Terminating the simulation

In the first case, with the press of an arrow, the user can spawn a maximum priority train (Priority = 5), arriving 30 seconds after the press and leaving 5 minutes after, with direction given by the arrow direction (Left or Right).
The other action is performed by pressing the key ”Space Bar”. All the threads will be terminated and the application closed.

4 Code Overview

4.1 Programming language and libraries

The application is entirely developed in C, using the libraries Allegro and pthread.
The project root directory has four sub-folders: conf contains the configuration files, docs the documentation, img the images used by the application and src, the most important one, stores all the source files. Two additional folders, bin and build, are created at compilation time and contain object and binary files, respectively.

4.2 Data structures

The state of the entities involved in the application is represented by few data structures. Here is a list of the most important ones:

- `train` (declared in train.h) is the train descriptor storing all the info about the current status and the behaviour of the different trains. Every train has its own
- `trains` (declared in train.h) is the array containing all the train descriptors
- cell (declared in station.h) is the cell descriptor storing all the info about the possible directions to which a train on that cell is available to move
- `busy_dock and busy_sec` (declared in station.h) are the arrays of booleans in which is stored the info about the status of the different docks and station's sectors. They are used to manage train scheduling in the station, to avoid collisions and to limit train's wait time

4.3 Tasks

Multiple real-time threads run concurrently during the simulation. Three threads are allocated for refreshing the graphics and handling keyboard. Also, each train has its own thread, which controls the movement according to the aforementioned model.

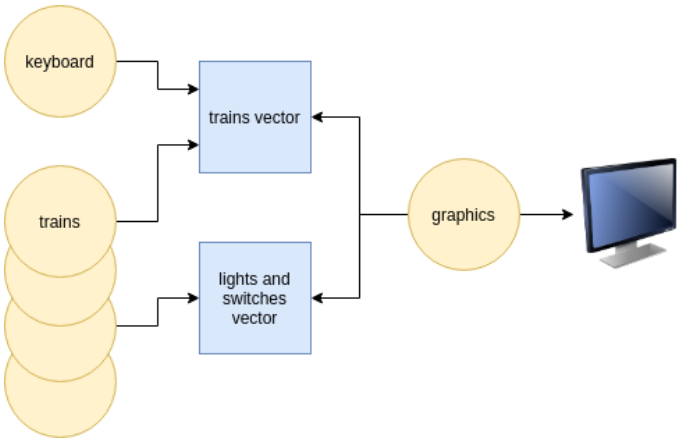

Figure 3: Task Interaction

- Keyboard task allows spawning a new train or terminating the simulation
- Graphics task updates the graphics
- Train tasks controls the motion and behaviour of the trains

The following table describes the default scheduling parameters:

	Period	Deadline	Priority
Keyboard	100	100	10
Graphics	20	20	50
Train	20	20	20