Giving precedence on the road

Andrea Princic

July 11, 2022

Giving precedence on the road

Report about the first Learning Unit for the course of Methods in Computer Science Education: Analysis.

Class

The Learning Unit is designed for students in the **second year of high school** (Liceo Scientifico delle Scienze Applicate).

Interdisciplinary objectives and motivations

The goal of this LU is to teach students the rule of **giving precedence** on the road, a topic in **Civic education**.
This may be useful to prepare students for the driving license test, since in Italy the minimum age to get a driving license (A1, B1) is 16 years old.
No prerequisites in Civic education are needed to take this LU except for the topic itself, explained in class.

Programming objectives and motivations

With respect to programming, the goal is to teach students the basic concepts of **block-based** programming and **parallel** programming in Scratch.
Prerequisites in Scratch are:

- variables
- conditionals
- loops
- blocks
- clones

Students will learn how to **start and synchronize parallel processes** by having to manage multiple cars in the scenario and having to choose which car has the precedence and which has to stop instead. They will have to figure out a way to make **cars "see" other cars** from other lanes.

Materials

No pre-programmed material will be given to students in order to prepare this LU. They will have to create **their own scenario** and figure out how to apply the rule in that scenario. The choice and implementation of an algorithm to make cars respect the rule is up to them.

Delivery

The LU will be delivered as a **homework** with a one-week deadline.

Evaluation

Criterion	Max score
Correct implementation of the rule in the chosen scenario	30
Use of parallelism	20
Detection algorithm accuracy and complexity	20
Scenario and rule application complexity	20
Graphics	10

Scores are scaled to 0–10.

Correct implementation of the rule in the chosen scenario

The most important criterion is the correct implementation of the rule. Since every student will submit a solution with a **personal scenario**, the correctness of the implementation will be evaluated w.r.t. the chosen scenario. Scenarios may include **traffic lights, signs, pedestrians and other objects** that may impact on the precedence on the road.

Use of parallelism

Use of parallelism plays an important role in the evaluation. Implementing a precedence rule requires running **multiple instances** of cars at the same time and to **synchronize** them in order to apply the rule.

Detection algorithm accuracy and complexity

A maximum of 2 points will be given for the accuracy and complexity of the detection algorithm. Points will be given based on the accuracy/complexity ratio. The accuracy of the algorithm will also be tested in the long run.

Scenario and rule application complexity

The more complex the chosen scenario is, the more difficult it will be to apply the rule. A scenario with a main **one-way road and one one-way secondary road** is the simplest scenario one can think of. A scenario with at least a **two-way road and at least two secondary roads** is required to get the maximum score.

Graphics

Graphics include the **smoothness in animations** and the **quality of graphics**.

Solutions

Best solution

The 10/10 solution proposes a scenario with a two-way main road and two secondary roads with a give precedence sign. Cars from secondary roads can **turn both directions** and this makes the application of the rule hard enough. Cars will have to look in one or both directions based on the whether they are going to turn right or left.

The algorithm that lets cars from a secondary road detect incoming cars from the main road is based on **danger zones** that react to the presence of cars above them. This lets cars from secondary roads detect incoming cars from the main road by looking for the state of said zones. The algorithm is very accurate since collisions between cars and danger zones are automatically computed by Scratch,

but it still has to differentiate between cars turning right and left: when turning right only on danger zone is to be considered, otherwise two.

Parallelism is very well managed in this scenario: cars from the main road spawn at given random intervals, while cars from the secondary roads only spawn when the previous car has already turned. This prevents cars from secondary roads to hit cars waiting to turn, while keeping quite random the number of cars that will be on the main road at a given moment.

Sufficient solution

The 6/10 solution proposes a one-way main road and one secondary road with a give precedence sign. Cars coming from the secondary road can only go straight on the main road. This is the simplest case possible for the task.

The algorithm is similar but of course simpler than the other, since there is only one danger zone. Cars won't have to differentiate the case of turning right or left because there is only one direction to go.

Here cars are all spawned at given random intervals, with cars from the secondary road spawning more seldom than those from the main road. This should be enough to avoid cars from the secondary road to hit cars waiting to turn, but is not 100% accurate.