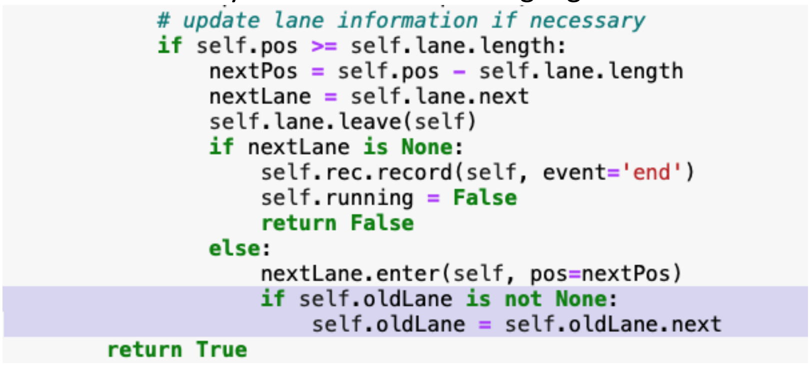
Multi Lane Mixed Simulation

Bug Fix 1

There is a bug that kicks in only when there is an end-of-lane condition triggered during the process of overtaking. Please replace/insert the lines highlighted below.

Class Vehicle method updateOnly:

At the very end insert the two highlighted lines

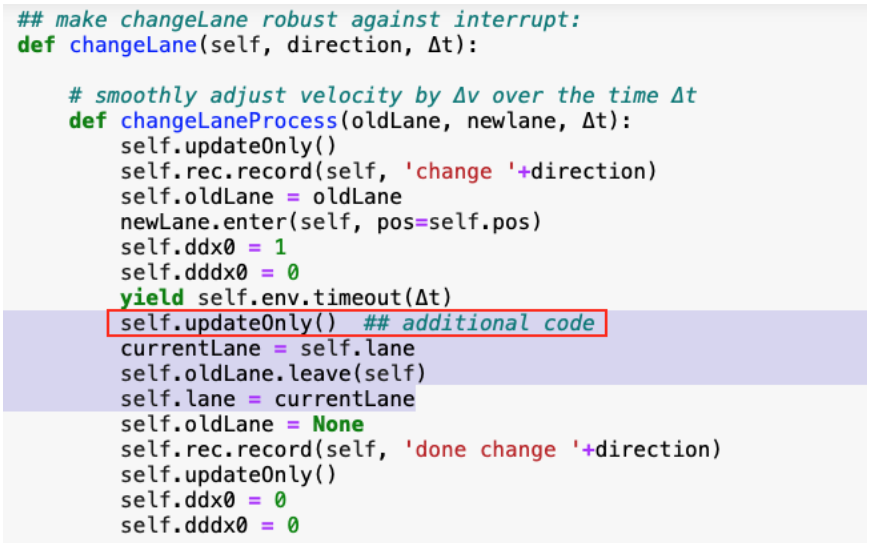


```
        # update lane information if necessary
        if self.pos >= self.lane.length:
            nextPos = self.pos - self.lane.length
            nextLane = self.lane.next
            self.lane.leave(self)
            if nextLane is None:
                self.rec.record(self, event='end')
                self.running = False
                return False
            else:
                nextLane.enter(self, pos=nextPos)
                if self.oldLane is not None:
                    self.oldLane = self.oldLane.next
    return True
```

Class Vehicle method changeLane

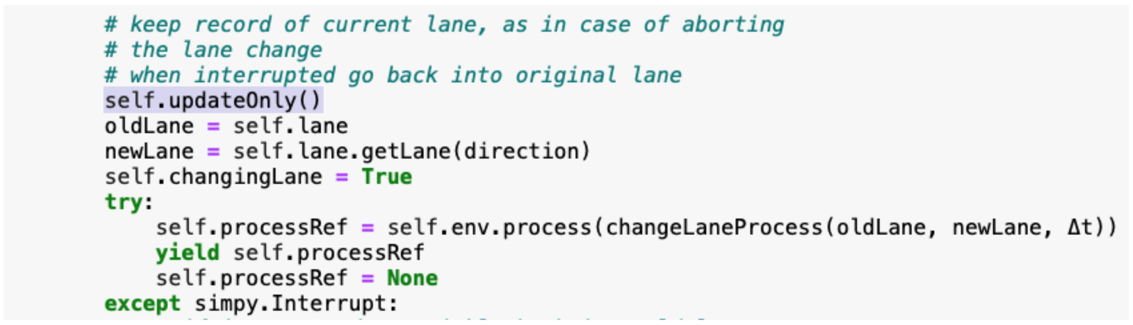
In class Vehicle, method changeLaneProcess:

The old code was just self.lane = newLane, which is in most cases ok, except when we ran during the overtaking process across an end of lane.



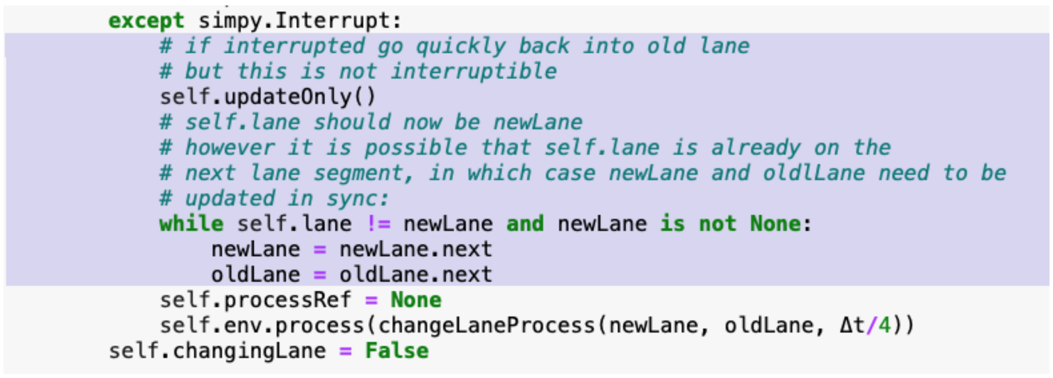
```
## make changeLane robust against interrupt:
def changeLane(self, direction, Δt):

    # smoothly adjust velocity by Δv over the time Δt
    def changeLaneProcess(oldLane, newlane, Δt):
        self.updateOnly()
        self.rec.record(self, 'change '+direction)
        self.oldLane = oldLane
        newLane.enter(self, pos=self.pos)
        self.ddx0 = 1
        self.dddx0 = 0
        yield self.env.timeout(Δt)
        self.updateOnly()  ## additional code
        currentLane = self.lane
        self.oldLane.leave(self)
        self.lane = currentLane
        self.oldLane = None
        self.rec.record(self, 'done change '+direction)
        self.updateOnly()
        self.ddx0 = 0
        self.dddx0 = 0
```

At the beginning of the body of changeLane(), i.e. just below the above code segment introduce updateOnly()



```
        # keep record of current lane, as in case of aborting
        # the lane change
        # when interrupted go back into original lane
        self.updateOnly()
        oldLane = self.lane
        newLane = self.lane.getLane(direction)
        self.changingLane = True
        try:
            self.processRef = self.env.process(changeLaneProcess(oldLane, newLane, Δt))
            yield self.processRef
            self.processRef = None
        except simpy.Interrupt:
```

And then just below that:



```
        except simpy.Interrupt:
            # if interrupted go quickly back into old lane
            # but this is not interruptible
            self.updateOnly()
            # self.lane should now be newLane
            # however it is possible that self.lane is already on the
            # next lane segment, in which case newLane and oldlLane need to be
            # updated in sync:
            while self.lane != newLane and newLane is not None:
                newLane = newLane.next
                oldLane = oldLane.next
            self.processRef = None
            self.env.process(changeLaneProcess(newLane, oldLane, Δt/4))
        self.changingLane = False
```

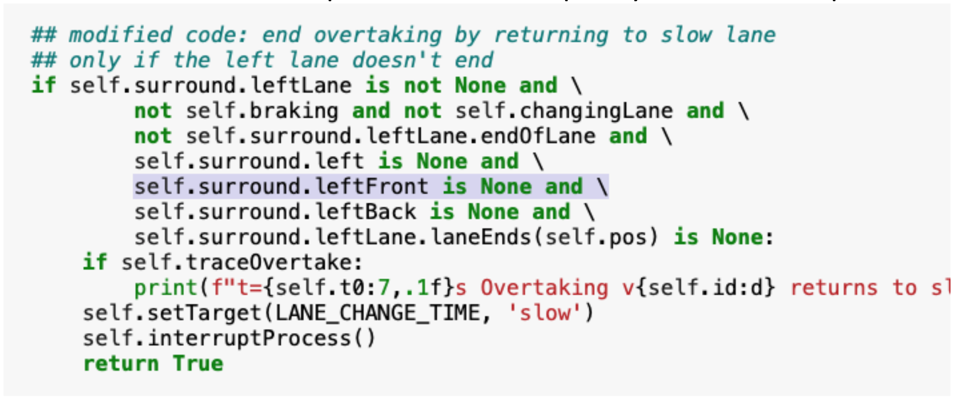
Bug Fix 2

Vehicles go quite aggressively back into the slower lane. This is relaxed, the cars should only switch into the slow lane if there is plenty of room ahead in the slow lane

Class Vehicle method update()

Class Vehicle at the end of update() replace the highlighted line as indicated:

The old code was just checking that the car in front on the left lane is not too slow. We now request that there is plenty of free head space.



```
  ## modified code: end overtaking by returning to slow lane
  ## only if the left lane doesn't end
  if self.surround.leftLane is not None and \
          not self.braking and not self.changingLane and \
          not self.surround.leftLane.endOfLane and \
          self.surround.left is None and \
          self.surround.leftFront is None and \
          self.surround.leftBack is None and \
          self.surround.leftLane.laneEnds(self.pos) is None:
      if self.traceOvertake:
          print(f"t={self.t0:7,.1f}s Overtaking v{self.id:d} returns to sl
      self.setTarget(LANE_CHANGE_TIME, 'slow')
      self.interruptProcess()
      return True
```