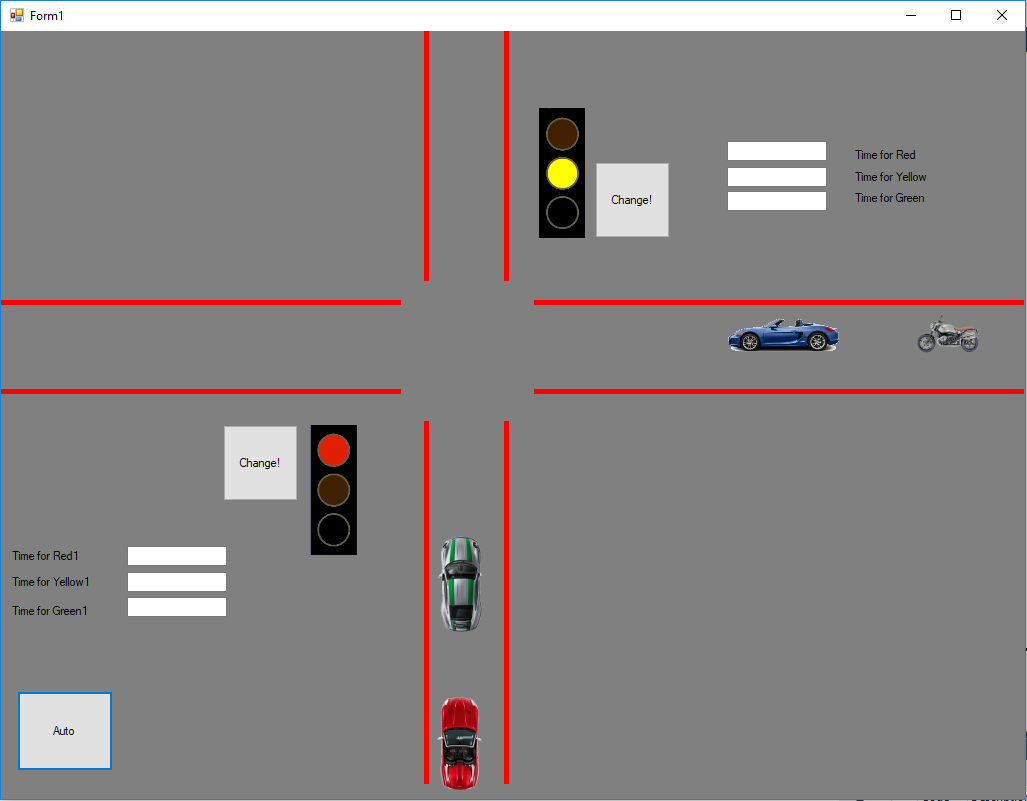
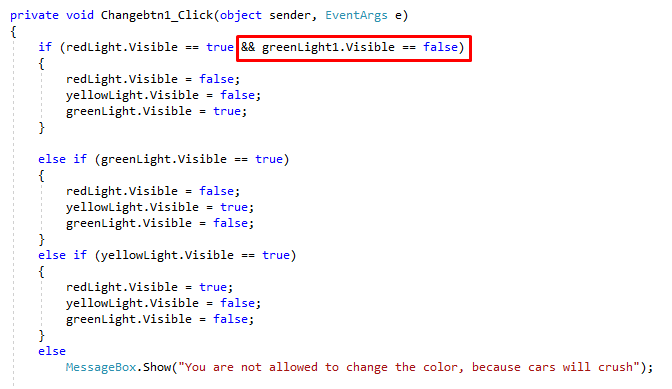
Το πρόγραμμα το οποίο μας αναθέσατε να κατασκευάσουμε είναι η κυκλοφορία των αμαξιών, και η διαχείρισή της με τα φανάρια. Η δικιά μου εφαρομή έχει την ακόλουθη εικόνα αφού τρέξετε (start) το πρόγρμμα.



Πριν την εκτέλεση η φόρμα έχει την ακόλουθη εικόνα, και αυτό γιατί γίνεται χρήση χρώματος, και δημιουργία γραμμών αφού εκτελεστεί το πρόγραμμα.

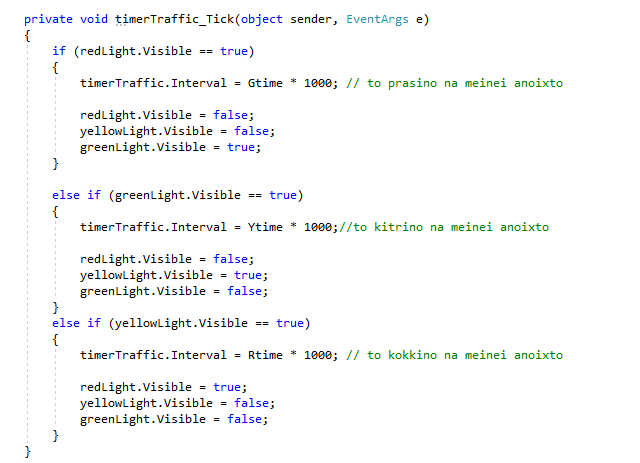


Για την κατασκευή του κουμπιού Change! χρησιμοποιήθηκε ο παρακάτω κώδικας, όπου κάθε φορά στο πάτημα του κουμπιού απο το χρήστη το φανάρι αλλάζει χρώμα. Αυτό γίνεται με ένα απλό έλεγχο (if) όπου κάθε φορά το χρώμα του φανάριου αλλάζει αναλόγως την συνθήκη (true ή false). Επίσης για τον ελέγχο να μην είναι και τα δύο φανάρια πράσινα υπάρχει η συνθήκη μέσα στο πλαίσιο, όπου απαγορέυει στο χρήστη να βάλει την ίδια χρονική στιγμή πράσινο, βγάζοντάς του το μήνυμα που είναι στο MessageBox.

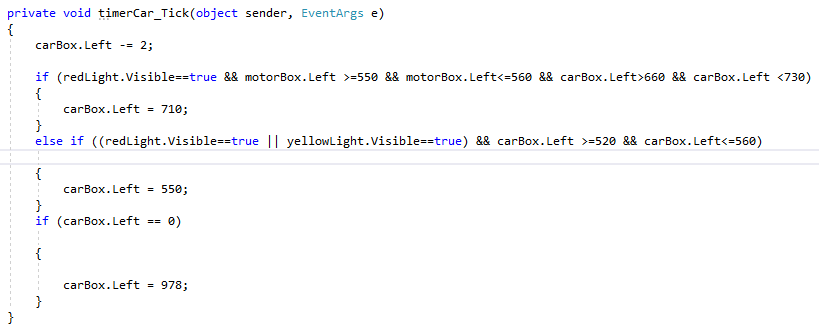


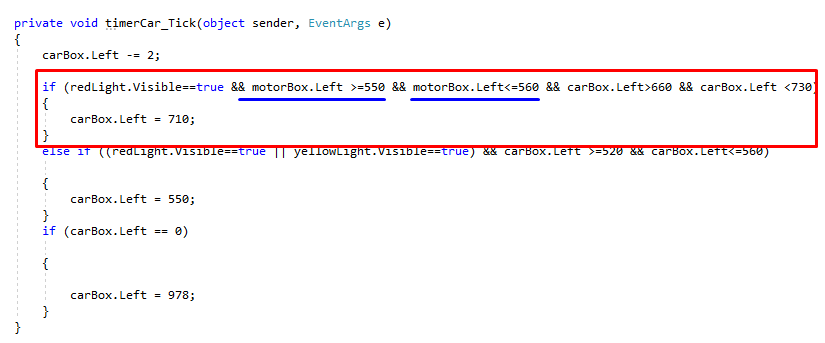
Τον ίδιο κώδικα ακολουθεί και το Changebtn2\_Click.

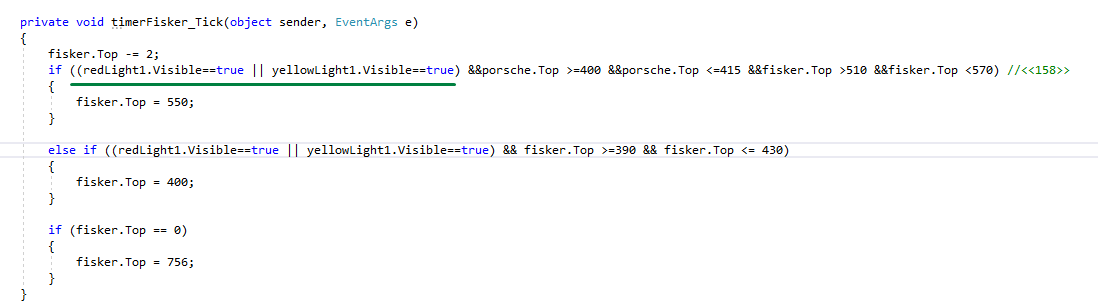
Για να επιτευχθεί η αλλαγή χρώματος στο φάναρι χωρίς την παρέμβαση του χρήστη κάθε φορά, παρά μόνο μία όπου χρειάζεται να δώσει τους χρόνους λειτουργίας του κάθε χρώματος, χρησιμοποίησα timer (timerTraffic η ονομασία του)όπου ακολουθεί ο παρακάτω κώδικας. Ο ίδιος κώδικας με timer χρησιμοποιείται και στο δεύτερο φανάρι (έχοντας το όνομα timerTraffic1). Ώστοσο για να καταφέρω να μετατρέψω τους αριθμούς που θα μου δώσει ο χρήστης στα textbox, πρώτα τα μετατρέπω σε ακέραιους με τη συνάρτηση int.TryParse και μετά αποθηκεύω τους χρόνους που μου έχει δώσει στις αντίστοιχες μεταβλητές (Rtime, Gtime, Ytime, R1time, G1time, Y1time). Σε κάθε συνθήκη if, else if , όπου ο timer αλλάζει το interval του βάση τις τιμές του χρήστη, κάνω την ανάλογη χρώματος μεταβλητή επι χίλια(\*1000), αυτό γιατί τα intervals των timers είναι σε milliseconds, και εγώ θέλω το χρόνο σε δευτερόλεπτα, επόμενως χρησιμοποιώ το πολλαπλασιασμό για την μετατροπή του.



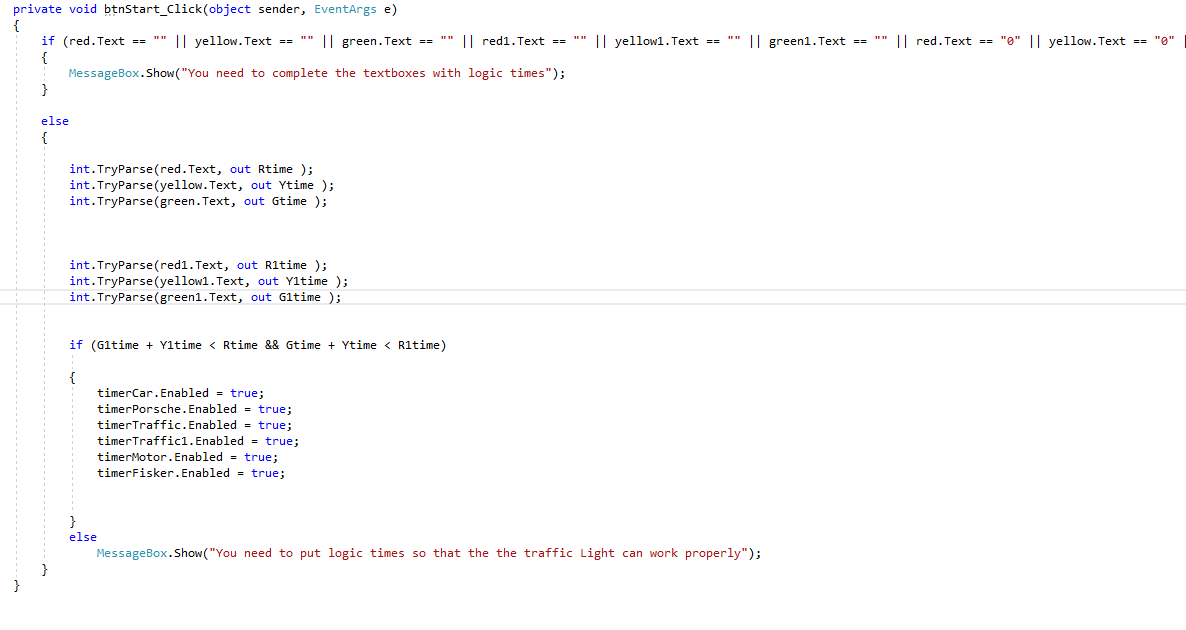
Η κίνηση των αμαξίων γίνεται με τη χρήση timers όπου σε κάθε tick τους μειώνουν την απόσταση τους από την θέση την οποία βρίσκονται προς την αρχή της φόρμας (δηλαδή το μπλέ αμάξι και η μηχανή κινούνται στον άξονα Χ και μειώνεται σε κάθε tick του timer -2 pixels προς το 0). Η ίδια ακριβώς λογική χρησιμοποιείται για την μετακίνηση του πράσινου και του κόκκινου αμαξιού(με την μόνη διαφορά ότι αντί για .Left, είναι .Top) ) .

Πχ. Ο timer του αμαξιού (που κινείται οριζόντια).

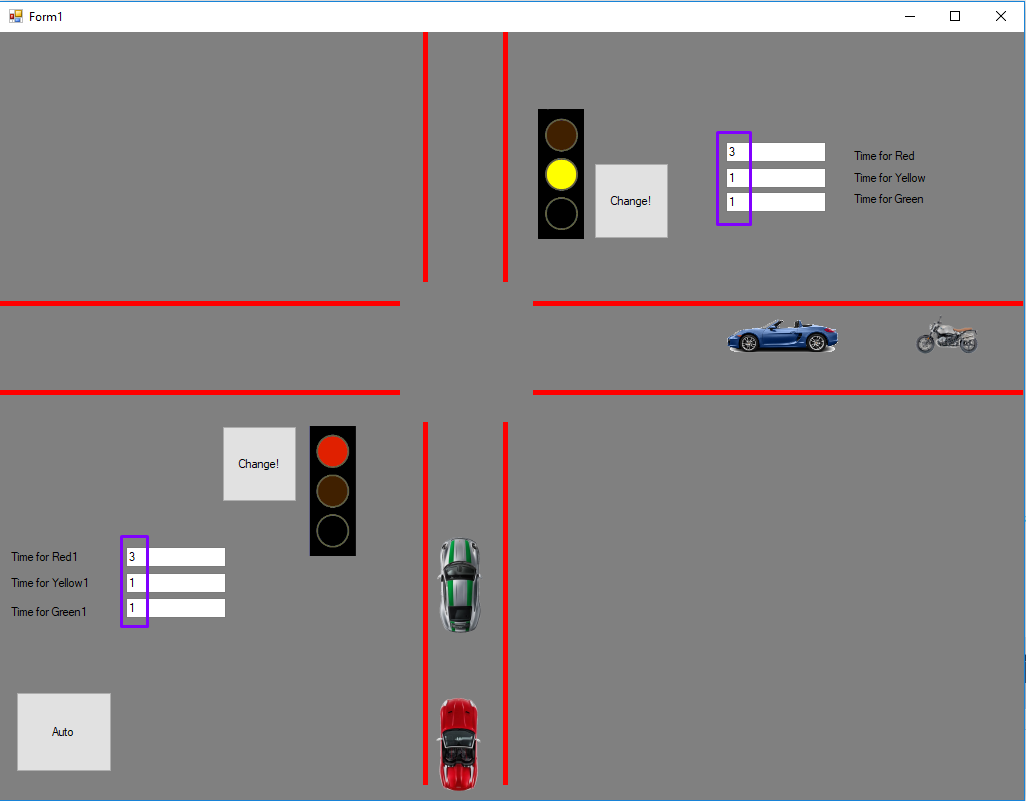
Για να μπορέσω, να σταματήσω το αμάξι κάθε φορά που έχει κόκκινο ή κίτρινο, ΟΧΙ όμως σε οποιδήποτε σημείο του «δρόμου», αλλά μόνο όταν είναι κοντά στη διασταύρωση, έθεσα όρια στο στη φόρμα (βλ.:carBox.Left >=660 && carBox.Left <=730 κτλ.). Έτσι μόνο όταν το αμάξι βρίσκεται εντός των ορίων και το φανάρι είναι κόκκινο ή κίτρινο μπορεί να σταματήσει. Να σημειώσω πως επειδή μας είπατε πως δεν σας ενδιαφέρει τι θα γίνεται στο κίτρινο, εγώ το πήρα με σκοπό να σταματάει. Μέσα στη ίδια συνθήκη έχω βάλει και το motorBox. Αυτό γίνεται για τον εξής λόγο.  Σε περιπτώση που το μηχανάκι σταματήσει και περάσει μόνο το αμάξι (κανονικός τρόπος κυκλοφορίας) τότε το αμάξι θα πρέπει να σταματήσει πίσω απο το μηχανάκι. Αν δεν είχα βάλει περιορισμούς για το μηχανάκι μέσα στη συνθήκη, τότε το αμάξι θα πήγαινε και θα σταματούσε πάνω του και θα έκρυβε την είκονα με το μηχανάκι, κάτι που δεν είναι λογικό. Την ίδια λογική χρησιμοποιώ και στο timerPorsche. Ακριβώς απο κάτω βρίσκεται η επόμενη συνθήκη όπου σταματάει τα αμάξια και στο κιτρίνο (όπως ανάφερα και πιο πάνω τα οχήματα σταμάτανε και στο κίτρινο φανάρι).

Στους timers motor και fisker χρησιμοποιώ ένα παραπάνω περιορισμό για το κίτρινο φανάρι, δίοτι χωρίς αυτό το περιορισμό θα έπεφταν πάνω στα μπροστινά αμάξια(το προηγούμενο πρόβλημα). 

Το επόμενο είναι η δημιουργία του κουμπιού Auto.



Στην παραπάνω απεικόνιση κώδικα είναι η δημιουργία του κουμπιού Auto. Αρχικά αναγκάζω στο χρήστη να βάλει τιμές στα φανάρια, διάφορες του μηδενός, αλλιώς δεν μπορει να συνεχίσει, γιατί και στην πραγματικότητα τα φανάρια έχουν κάποιο χρόνο λειτουργίας, αλλιώς δεν υφίσταται η χρήση τους. Εφόσον ο χρήστης δώσει τους χρόνους γίνεται η μετατροπή απο String σε Int και ακριβώς απο κατώ ακολουθεί ο κώδικας ελέγχου έγκυρων τιμών. Έγκυρων τιμώ εννοώ τις τιμές που αναγκάζω το χρήστη να βάλει κάθε φορά για να δουλέψει σωστά το φανάρι. Εν ολίγης το άθροισμα του πράσινου και του πορτοκαλί φαναριού1 πρέπει να είναι μικρότερο απο το χρόνο του κόκκινου φαναριού2, το οποίο είναι πολύ λογικό, διότι, δεν γίνεται όσο το φανάρι1 είναι πράσινο, τότε να μπορεί και το φανάρι2 να γίνει πράσινο. Επόμενς το άθροισμα των 2 θα πρέπει να ισούται με το κόκκινο του άλλου για να μπορεί να δουλέψει η κυκλοφορία σωστά και να μην δημιουργηθεί κάποιο τρακάρισμα.

*Προτεινόμενοι χρόνοι για την εξακρίβωση του συγχρονισμού είναι****: ***

*Βέβαια μπορείτε να χρησιμοποιήσετε όποιους χρόνους επιθυμήτε, απλά προτείνω τους συγκρκριμένους γιατί, είναι γρήγορος και ορατός ο συγχρονισμός.*

**Σημειώνεται πως για την εύρεση των ακριβές συντεταγμένων χρησιμοποιηθήκε ο κώδικας που είχαμε διδαχθεί στο μάθημα (coordinates), ο οποίος έχει αφαιρεθεί.**