ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Χειμερινό εξάμηνο ακ. έτους 2025-2026
Εργασία μαθήματος*

Η παρούσα εργασία έχει ως αντικείμενο την απλοποιημένη προσομοίωση λειτουργίας και πλοήγησης (βάσει GPS) ενός πλήρως αυτόνομου οχήματος (full self-driving car) σε έναν κόσμο-πλέγμα. Το σύστημα θα πρέπει να επιδεικνύει τις αρχές του αντικειμενοστραφούς προγραμματισμού, συμπεριλαμβανομένων του εγκλωβισμού (encapsulation), της κληρονομικότητας (inheritance), του πολυμορφισμού (polymorphism), της σύνθεσης (composition) και της επικοινωνίας μεταξύ αντικειμένων (messaging). Η υλοποίηση να γίνει στην C++.

Σε διάφορες ενότητες της εκφώνησης, παρέχονται υποδείξεις/προτάσεις υλοποίησης, αλλά έχετε τη δυνατότητα να προσθέσετε «περιπλοκότητα» για να γίνει πιο ενδιαφέρουσα η προσομοίωση. Σε κάθε περίπτωση, τεκμηριώστε τις επιλογές σας στο τελικό παραδοτέο.

1 Περιγραφή του συστήματος

Ο Κόσμος-Πλέγμα (**GridWorld**) είναι ένα δισδιάστατο περιβάλλον πλέγματος συγκεκριμένων διαστάσεων (πχ. 40×40 θέσεις/κελιά). Το αυτόνομο όχημα (**SelfDrivingCar**) περιλαμβάνει ένα σύνολο αισθητήρων (**Sensors**) και ένα σύστημα πλοήγησης (**NavigationSystem**), καθώς και ένα σύνολο εντολών πλοήγησης που πρέπει να ακολουθήσει. Το σύστημα πλοήγησης παίρνει αποφάσεις βάσει των δεδομένων που του επιστρέφει η μηχανή συγχώνευσης αισθητήρων (**SensorFusionEngine**). Μια πληθώρα αντικειμένων (**WorldObject**) προσομοιώνονται και τοποθετούνται σε τυχαία κελιά του κόσμου. Αυτά, εν τέλει, αποτελούν πηγή δεδομένων για το σύστημα πλοήγησης και τη μηχανή συγχώνευσης αισθητήρων του. Οι αισθητήρες αναγνωρίζουν διαφορετικά αντικείμενα και αντλούν διαφορετικές πληροφορίες από τα αντικείμενα, τις οποίες παρέχουν στη μηχανή συγχώνευσης αισθητήρων ώστε να παραχθεί μια ενοποιημένη εικόνα του κόσμου. Αυτά τα δεδομένα τα επεξεργάζεται το σύστημα πλοήγησης, και εν τέλει καθορίζει τη συμπεριφορά του αυτόνομου οχήματος καθώς ακολουθεί την πορεία του.

Για τη προσομοίωση, δημιουργείται ο Κόσμος και τοποθετείται πλήθος αντικειμένων σε τυχαίες θέσεις/κελιά (**Position**), καθώς και το αυτόνομο όχημα. Ο χρήστης έχει τη δυνατότητα να καθορίσει το φύτρο (**seed**) της γεννήτριας τυχαίων

*Για την ανάπτυξη και την σύνταξη της εκφώνησης καθώς και για την υλοποίηση ευχαριστώ θερμά τον υπ. Διδ. κύριο Βασίλειο-Μάριο Ανατασίου

αριθμών από τη γραμμή εντολών (σε αντίθετη περίπτωση χρησιμοποιείται ο τωρινός χρόνος), καθώς και το πλήθος των διακριτών χρονικών βημάτων (**ticks**) για το οποίο θα τρέχει η προσομοίωση. Σε κάθε βήμα, ανανεώνεται (**update**) η κατάσταση και θέση των αντικειμένων του κόσμου, και έπειτα ανανεώνεται η κατάσταση του αυτόνομου οχήματος.

1.1 Βήματα προσομοίωσης

Η προσομοίωση εκτελείται σε διακριτά βήματα, όπου κάθε βήμα αντιπροσωπεύει μια αυθαίρετη μονάδα χρόνου. Όλα τα αντικείμενα ενημερώνουν την κατάστασή τους σε κάθε βήμα: τα κινούμενα οχήματα μετακινούνται βάσει ταχύτητας/κατεύθυνσης, τα φανάρια ενημερώνουν την εσωτερική τους κατάσταση βάσει του βήματος προσομοίωσης, ενώ το όχημα συλλέγει δεδομένα αισθητήρων, τα συγχωνεύει, ενημερώνει την ταχύτητά του, και κινείται βάσει των στόχων GPS.

Οι παράμετροι της προσομοίωσης, όπως πχ. το πλήθος των διαφορετικών αντικειμένων, θα έχει προκαθορισμένες τιμές, αλλά όμως θα μπορούν να επανακαθοριστούν από τη γραμμή εντολών. Για την παραμετροποίηση, να ελέγχεται αν παρέχονται αντίστοιχα ζευγάρια παραμέτρων-τιμών, για παράδειγμα --seed 12, --simulationTicks 50, --numStopSigns 4 κ.ο.κ. Οι στόχοι GPS θα παρέχονται πάντα στο τέλος των παραμέτρων ως --gps <x1> <y1> <x2> <y2>... Πρέπει να παρέχονται ένας ή παραπάνω στόχοι.

Η προσομοίωση τελειώνει όταν εκτελεστούν τα απαιτούμενα βήματα ή αν το αυτόνομο όχημα προσπαθήσει να κινηθεί εκτός των ορίων του κόσμου (ανεξάρτητα από τα άλλα κινούμενα οχήματα).

Πρέπει να υποστηρίζεται και η παράμετρος --help που θα εκτυπώνει οδηγίες χρήσης του προγράμματος. Παρέχεται ενδεικτική συνάρτηση:

```
void printHelp()
{
    cout << "Self-Driving Car Simulation" << endl;
    cout << "Usage: " << endl;
    cout << "  --seed <n>                    Random seed (default: current time)" << endl;
    cout << "  --dimX <n>                    World width (default: 40)" << endl;
    cout << "  --dimY <n>                    World height (default: 40)" << endl;
    cout << "  --numMovingCars <n>           Number of moving cars (default: 3)" << endl;
    cout << "  --numMovingBikes <n>          Number of moving bikes (default: 4)" << endl;
    cout << "  --numParkedCars <n>           Number of parked cars (default: 5)" << endl;
    cout << "  --numStopSigns <n>            Number of stop signs (default: 2)" << endl;
    cout << "  --numTrafficLights <n>        Number of traffic lights (default: 2)" << endl;
    cout << "  --simulationTicks <n>         Maximum simulation ticks (default: 100)" << endl;
    cout << "  --minConfidenceThreshold <n> Minimum confidence cutoff (default: 40)" << endl;
    cout << "  --gps <x1> <y1> [x2 y2 ...]  GPS target coordinates (required)" << endl;
    cout << "  --help                        Show this help message" << endl << endl;
    cout << "Example usage:" << endl;
    cout << "  ./oopproj_2025 --seed 12 --dimY 50 --gps 10 20 32 15" << endl;
}
```

1.2 Κατηγορίες αντικειμένων

Στον κόσμο δημιουργούνται και τοποθετούνται αντικείμενα διαφόρων κατηγοριών:

- Κινούμενα αντικείμενα (**MovingObjects**), όπως άλλα αυτοκίνητα ή ποδήλατα, και

- Στατικά αντικείμενα (**StaticObjects**), που έχουν εσωτερική κατάσταση αλλά δε μετακινούνται.

Κάθε αντικείμενο του κόσμου έχει μια αλφαριθμητική ταυτότητα (**ID**) καθώς και ένα χαρακτήρα αναπαράστασης (**glyph**) που χρησιμοποιείται για απεικόνιση κατά την οπτικοποίηση (λεπτομέρειες για την οπτικοποίηση δείτε στο τέλος στη σχετική ενότητα). Η ταυτότητα προκύπτει από το όνομα της κατηγορίας που ανήκει το αντικείμενο, μαζί με έναν αύξοντα αριθμό ανά κατηγορία (π.χ. BIKE:4) ώστε να αναγνωρίζεται εύκολα στα μηνύματα εξόδου.

1.2.1 Στατικά αντικείμενα

Τα στατικά αντικείμενα μπορεί να είναι:

- Στάσιμα οχήματα (**StationaryVehicles**), που είναι απλά σταθμευμένα αυτοκίνητα.
- Οδικές πινακίδες (**TrafficSigns**), όπως πινακίδες STOP, πινακίδες ορίου ταχύτητας, τοπωνύμια, απαγορευτικά, ή άλλες πινακίδες.
- Φανάρια (**TrafficLights**).

Τα στάσιμα οχήματα δεν έχουν κάποια οδηγική συμπεριφορά, δημιουργούνται σε μια θέση και παραμένουν εκεί. Επίσης, δεν επηρεάζουν την οδική συμπεριφορά του οχήματος.

Οι οδικές πινακίδες περιγράφονται από το κείμενό τους (δηλαδή τη πληροφορία που μεταδίδουν). Στα πλαίσια της προσομοίωσης, μόνο η πινακίδα STOP επηρεάζει την οδική συμπεριφορά.[1]

Τα φανάρια έχουν εσωτερική κατάσταση, που είναι μια εκ των ΚΟΚΚΙΝΟ (RED), ΚΙΤΡΙΝΟ (YELLOW), ΠΡΑΣΙΝΟ (GREEN), και μια αίσθηση του χρόνου από την αρχή της προσομοίωσης (**ticks**). Ο κύκλος ενός φαναριού είναι ΚΟΚΚΙΝΟ (διάρκεια: 4 ticks) → ΠΡΑΣΙΝΟ (διάρκεια: 8 ticks) → ΚΙΤΡΙΝΟ (διάρκεια: 2 ticks) → ΚΟΚΚΙΝΟ. Μπορείτε να θεωρήσετε πως όλα ξεκινάνε με ΚΟΚΚΙΝΟ ως απλοποίηση, αλλά ιδανικά θα ξεκινάνε σε μια τυχαία κατάσταση.

1.2.2 Κινούμενα Αντικείμενα

Τα κινούμενα αντικείμενα έχουν μια σταθερή ταχύτητα ανά κατηγορία (**speed**), και μετακινούνται διαρκώς σε μια κατεύθυνση (**direction**) που επιλέγεται τυχαία κατά τη δημιουργία τους. Η αναπαράσταση της κατεύθυνσης στους 4 ορίζοντες αφήνεται στην ευχέρειά σας. Αν ένα κινούμενο όχημα προσπαθήσει να μετακινηθεί εκτός ορίου του κόσμου, αφαιρείται από την προσομοίωση. Δε χρειάζεται τα κινούμενα οχήματα (αυτοκίνητα ή ποδήλατα) να έχουν «έξυπνη» συμπεριφορά ή να αλλάζουν την ταχύτητά τους, πχ. βάσει φαναριών που βρίσκουν μπροστά τους.

[1] Προτείνεται να προσθέσετε και άλλους τύπους (πχ. YIELD) με πιθανή επιρροή στο αυτόνομο όχημα.

1.3 Αυτόνομο όχημα

Το αυτόνομο όχημα τοποθετείται σε ένα κελί και έχει κατεύθυνση (**direction**) και ταχύτητα (**speed**). Οι καταστάσεις ταχύτητας είναι **STOPPED** (σταματημένο), **HALF_SPEED** (το ήμισυ της πλήρους ταχύτητας) ή **FULL_SPEED** (2 θέσεις ανά tick).

Όταν το όχημα επιταχύνει (**accelerate**), πάει από **STOPPED** σε **HALF_SPEED**, ή από **HALF_SPEED** σε **FULL_SPEED**. Επιβραδύνει (**decelerate**) όταν εντοπίζει κίτρινο ή κόκκινο φανάρι μπροστά εντός 3 θέσεων, όταν πλησιάζει στο στόχο GPS εντός 5 θέσεων, ή όταν βρίσκει εμπόδιο μπροστά (δηλαδή κινούμενο αντικείμενο) εντός 2 θέσεων. Επίσης, εκτελεί περιστροφή (**turn**) όταν πρέπει να αλλάξει κατεύθυνση.

Το όχημα έχει ένα πλήθος αισθητήρων (**Sensors**) -αναλυτική περιγραφή τους δίνεται παρακάτω. Στην τυπική περίπτωση, έχει έναν αισθητήρα από κάθε είδος, αλλά δύναται να έχει μεταβλητό πλήθος από οποιοδήποτε υποσύνολο αισθητήρων. Το όχημα θα συλλέξει μετρήσεις από όλους τους αισθητήρες (**collectSensorData**), θα τις μεταφέρει στο σύστημα πλοήγησης (**syncNavigationSystem**), και θα λάβει απόφαση μετακίνησης από το σύστημα πλοήγησης ώστε να προχωρήσει κατάλληλα (**executeMovement**).

1.3.1 Σύστημα Πλοήγησης

Το αυτόνομο όχημα έχει ένα σύστημα πλοήγησης (**NavigationSystem**). Στην αρχή της προσομοίωσης, το σύστημα πλοήγησης λαμβάνει το σύνολο συντεταγμένων/στόχων που πρέπει να ακολουθήσει. Για την αναπαράσταση των συντεταγμένων/στόχων μπορείτε να χρησιμοποιήσετε πλειάδες (tuples, πχ. (<x1>,<y1>)), κελιά (πχ. Position(<x1>,<y1>)), ή ζεύγη συντεταγμένων (πχ.<x1> <y1>), όπως προτιμάτε.

Όταν το αυτόνομο όχημα παρέχει τις νέες μετρήσεις αισθητήρων στο σύστημα πλοήγησης, αυτό πρέπει με τη σειρά του να τις συγχωνεύσει. Για αυτόν το σκοπό χρησιμοποιεί τη μηχανή συγχώνευσης αισθητήρων (**SensorFusionEngine**), η οποία αξιοποιεί τα νέα δεδομένα και εκτελεί τη λειτουργία συγχώνευσης (**fuseSensorData**). Βάσει των επιμέρους μετρήσεων, ανανεώνει την εσωτερική του μνήμη/κατάσταση με μια συλλογή ενοποιημένων μετρήσεων αισθητήρων. Έτσι, είναι πλέον έτοιμο να αναλογιστεί τα δεδομένα του κόσμου και να πάρει απόφαση πλοήγησης (**makeDecision**) βάσει του τωρινού στόχου.

Μπορείτε να ορίσετε την κατεύθυνση μετακίνησης βάσει του επόμενου στόχου όπως επιθυμείτε, πχ. να πηγαίνει το όχημα πρώτα στον κατάλληλο κατακόρυφο άξονα και μετά στον οριζόντιο, ή πρώτα στο σωστό «ύψος» και μετά «πλάτος». Οι αποστάσεις μεταξύ κελιών ορίζονται απλά με απόσταση Manhattan/Taxicab. Για παράδειγμα, οι θέσεις (3,3) και (5,7) απέχουν |5-3|+|7-3|=6. Για να μεταφερθεί από την πρώτη στη δεύτερη, το όχημα χρειάζεται να μετακινηθεί 2 θέσεις οριζόντια και 4 κατακόρυφα. Ορίστε το σημείο (0,0) όπου επιθυμείτε (πχ. κάτω αριστερά στο GridWorld ίσως είναι η απλούστερη λύση).

2 Αναγνώριση αντικειμένων

2.1 Περιγραφή

Κάθε αισθητήρας έχει εμβέλεια, οπτικό πεδίο, καθώς και βασική/λειτουργική ακρίβεια για απόσταση και αναγνώριση κατηγορίας αντικειμένου. Κάθε αισθητήρας ανιχνεύει είτε στατικά είτε/και κινούμενα αντικείμενα ανάλογα την κατηγορία του. Οι αισθητήρες είναι εγκατεστημένοι πάνω στο όχημα, επομένως λαμβάνουν τη θέση και την κατεύθυνση του οχήματος. Ακολουθεί αναλυτική περιγραφή των αισθητήρων και της λειτουργικότητάς τους.

Κάθε αισθητήρας επιστρέφει μια συλλογή δεδομένων (**SensorReading**) που περιγράφει τα χαρακτηριστικά της μέτρησης, όπως κατηγορία αντικειμένου (**type**), απόσταση (**distance**), θέση (**position**), ταυτότητα (**objectId**), βεβαιότητα (**confidence**), ταχύτητα (**speed**), κατεύθυνση (**direction**), κείμενο πινακίδας (**signText**), χρώμα φωτός (**trafficLight**). Δεν υποστηρίζονται όλες οι λειτουργίες από κάθε αισθητήρα, ούτε παρέχονται όλες οι πληροφορίες από κάθε αντικείμενο που προσμετράται από όλους τους αισθητήρες.

2.2 Τύποι αισθητήρων

2.2.1 Αισθητήρας Lidar (Lidar Sensor)

- Εμβέλεια: 9 κελιά
- Οπτικό πεδίο: Ένα τετράγωνο 81 κελιών, με κεντρικό κελί στο όχημα (δηλαδή όραση «360 μοιρών» 9x9 κελιών)
- Ανιχνεύει: Όλα τα αντικείμενα (στατικά και κινούμενα), όχι όμως χρώμα φαναριών ή κείμενο πινακίδας
- Επιστρέφει: Απόσταση, κατηγορία αντικειμένου, βεβαιότητα
- Ακρίβεια: Εξαιρετική για απόσταση, μέτρια για κατηγοριοποίηση

2.2.2 Αισθητήρας ραντάρ (Radar Sensor)

- Εμβέλεια: 12 κελιά
- Οπτικό πεδίο: 12 κελιά ευθεία μπροστά
- Ανιχνεύει: Μόνο κινούμενα αντικείμενα
- Επιστρέφει: Απόσταση, ταχύτητα, κατεύθυνση κίνησης, βεβαιότητα
- Ακρίβεια: Υψηλή για απόσταση και για κατηγοριοποίηση

2.2.3 Αισθητήρας κάμερας (Camera Sensor)

- Εμβέλεια: 7 κελιά
- Οπτικό πεδίο: Ένα τετράγωνο 7x7 (49 κελιά) ακριβώς μπροστά από το όχημα

- Ανιχνεύει: Όλα τα αντικείμενα, συμπεριλαμβανομένων χρώματος φαναριών και κειμένου πινακίδων
- Επιστρέφει: Όλες τις δυνατές τιμές
- Ακρίβεια: Υψηλή για κατηγοριοποίηση, μέτρια για απόσταση

Στις παραπάνω περιγραφές, η ακρίβεια ενδεικτικά μεταφράζεται στα παρακάτω ποσοστά:

- Εξαιρετική: 99%
- Υψηλή: 95%
- Μέτρια: 87%
- Χαμηλή: 75%

Προτείνεται να πειραματιστείτε με τιμές που οδηγούν σε πιο ενδιαφέρουσα προσομοίωση.

2.2.4 Βεβαιότητα αισθητήρων

Σε κάθε ανάγνωση, ο αισθητήρας βρίσκει το αναγνωρίσιμο σύνολο αντικειμένων του κόσμου, με κάποιο βαθμό βεβαιότητας (confidence, από 0.0 έως 1.0), που επηρεάζεται από την ακρίβεια των μετρήσεων.

Η βεβαιότητα καθορίζεται από διάφορους παράγοντες, οι οποίοι επηρεάζουν το βασικό ποσοστό ακρίβειας του αισθητήρα. Λεπτομερώς:

- **Τύπος αισθητήρα**: Κάθε αισθητήρας έχει ένα βασικό ποσοστό ακρίβειας: Εξαιρετικό για Lidar και Radar, Υψηλό για κάμερα.
- **Απόσταση**: Εκτός εμβέλειας δεν αναγνωρίζονται καν αντικείμενα, στο όριο της εμβέλειας αναγνωρίζονται με μηδαμινή βεβαιότητα, και με γραμμικό τρόπο όσο κοντύτερα, τόσο υψηλότερη η βεβαιότητα.
- **Κατηγορία αντικειμένου**: Μερικοί αισθητήρες είναι καλύτεροι στην ανίχνευση συγκεκριμένων κατηγοριών (πχ. εξαιρετική αναγνώριση φαναριών και οχημάτων, υψηλή αναγνώριση ποδηλάτων κ.ο.κ.).
- **Τυχαία μεταβολή**: Κάθε ανάγνωση, ανεξαρτήτως κατηγορίας, έχει ±0.05 θόρυβο (noise) για προσομοίωση ανακρίβειας του πραγματικού κόσμου.

Δε θα καθορίσουμε λεπτομερή φόρμουλα στα πλαίσια της εκφώνησης, εκτός από τους παραπάνω ενδεικτικούς κανόνες.

3 Μηχανή συγχώνευσης αισθητήρων

Το όχημα έχει αισθητήρες που παρέχουν διαρκώς πληροφορίες για το περιβάλλον και την εσωτερική κατάστασή του και τις οποίες πρέπει να «φιλτράρει» και να συνθέσει ούτως ώστε να εκπληρώσει το στόχο μετάβασής του.[2]

Επειδή κάθε αισθητήρας αναγνωρίζει διαφορετικές κατηγορίες αντικειμένων με διαφορετικό βαθμό ακρίβειας, είναι απαραίτητο να έχουμε ένα σύστημα που συγχωνεύει και τυποποιεί τις διαφορετικές μετρήσεις. Κάθε αισθητήρας παράγει μια ομαδοποίηση των μετρήσεών του για κάθε αντικείμενο που αναγνωρίζει. Βάσει των αντίστοιχων προηγούμενων περιγραφών, οι μετρήσεις κάποιων αισθητήρων είναι πιο πλήρεις από άλλων. Η μηχανή συγχώνευσης αισθητήρων λαμβάνει το σύνολο των ομαδοποιημένων μετρήσεων και τις μετατρέπει σε μια τυποποιημένη ανάγνωση μετρήσεων (**SensorReading**), συμπληρώνοντας τυχόν ανύπαρκτες μετρήσεις. Για παράδειγμα, μόνο η κάμερα θα αναγνωρίζει το κείμενο μιας επιγραφής ή το χρώμα του φαναριού, ενώ τα radar/lidar αναγνωρίζουν κινούμενα αντικείμενα με μεγαλύτερη ακρίβεια. Όποια τιμή είναι άγνωστη να λαμβάνει κατάλληλη τιμή (πχ. 0.0, -1, «N/A» κ.ο.κ.). Εν τέλει, η μηχανή επιστρέφει μία ενιαία μέτρηση για κάθε αντικείμενο που ανιχνεύτηκε με επαρκή βεβαιότητα. Θα περιγράψουμε τη λειτουργία της συνοπτικά. Στα πλαίσια της προσομοίωσης αρκεί μια απλοϊκή υλοποίηση μέσου όρου.

Η υλοποίηση απαιτεί μια μέθοδο (**fuseSensorData**) που θα λαμβάνει πολλαπλές μετρήσεις για όλα τα αντικείμενα, θα ενοποιεί αυτές που αφορούν κοινά αντικείμενα, θα υπολογίζει συγχωνευμένες μετρήσεις, και θα επιστρέφει ένα νέο μικρότερο σύνολο μετρήσεων με επαρκή βεβαιότητα. Όπως είναι φυσικό, με έναν μόνο αισθητήρα, δε γίνεται οποιαδήποτε περαιτέρω επεξεργασία/συγχώνευση.

Για να επιτευχθεί αυτή η ενοποιημένη αντίληψη, αρχικά συγχωνεύονται ανιχνεύσεις βάσει του ID του αντικειμένου (δηλαδή, συλλέγουμε όλες τις μετρήσεις που αφορούν το κάθε αντικείμενο με ένα συγκεκριμένο ID ώστε να τις συγχωνεύσουμε). Με τη χρήση σταθμισμένου μέσου όρου (weighted average), οι αναγνώσεις υψηλότερης βεβαιότητας έχουν μεγαλύτερη επιρροή.

Η μηχανή έχει ένα κατώφλι βεβαιότητας (**minConfidenceThreshold**) ώστε να απορρίψει μετρήσεις χαμηλής βεβαιότητας. Για λόγους ασφαλείας όμως, αν έστω και ένας αισθητήρας αναγνωρίσει ποδήλατο, η μέτρηση δεν απορρίπτεται. Το κατώφλι βεβαιότητας αποτελεί παράμετρο της προσομοίωσης.

Για μια ενδιαφέρουσα επισκόπηση σύγχρονων συστημάτων συγχώνευσης δεδομένων, μπορείτε να δείτε το σύνδεσμο στην υποσημείωση.[3]

4 Οπτικοποίηση (Visualization)

Προκειμένου να αναδειχθεί η εκτέλεση της προσομοίωσης, να υλοποιηθούν δύο τύποι οπτικοποίησης:

Η πλήρης (**visualization_full**) οπτικοποίηση θα εκτελείται στην αρχή και στο τέλος της προσομοίωσης, και θα εκτυπώνει τον πλήρη κόσμο σε πλέγμα.

Η μερική (**visualization_pov**) οπτικοποίηση θα εκτυπώνεται σε κάθε βήμα, και θα αφορά το κομμάτι του κόσμου που βρίσκεται γύρω από το όχημα. Θα έχει

[2]Extended Confidence-Weighted Averaging in Sensor Fusion

[3]MDPI - Sensors (Volume 25, Issue19) - A Review of Multi-Sensor Fusion in Autonomous Driving

μεταβλητή ακτίνα (**radius**), και θα μπορεί να εκτυπώνει είτε τα κελιά τριγύρω από το όχημα (centered) είτε τα κελιά μπροστά στο οπτικό του πεδίο (front).

4.1 Απεικόνιση και προτεραιότητα

Καθώς μπορεί να υπάρχουν πολλά διαφορετικά αντικείμενα σε κάθε κελί του κόσμου, παρέχεται η λίστα απεικονίσεων κατά προτεραιότητα:

Self-Driving Car	→ @
Empty Cell	→ .
Outside Bounds	→ X
RED Light	→ R
YELLOW Light	→ Y
Stop Sign	→ S
Moving Bike	→ B
Moving Car	→ C
GREEN Light	→ G
Parked Car	→ P
Unknown/Other Object	→ ?

4.2 Εκτυπώσεις (logging)

Στα πλαίσια του παραδοτέου, δεν απαιτείται να εκτυπώνεται κείμενο όταν εκτελούνται διάφορες ενέργειες (πχ. δημιουργία/καταστροφή αντικειμένων, ανίχνευση, μετακίνηση κλπ). Βεβαίως, προτείνεται να υλοποιήσετε εκτυπώσεις τόσο για λόγους ευκολίας αποσφαλμάτωσης όσο και «ζωντάνιας» της προσομοίωσης. Μπορείτε, αν το επιθυμείτε, αντί να εκτυπώνετε στην οθόνη, να γράφετε τα μηνύματα σε ένα *oopproj_2025.log* αρχείο.

Παρέχεται λίστα ενδεικτικών μηνυμάτων constructors (+)/destructors (-) προς διευκόλυνσή σας:

```
[+WORLD:<id>] Reticulating splines - Hello, world!
[-WORLD:<id>] Goodbye, cruel world!
[+LIDAR:<id>] Lidar sensor ready - Sensing with pew pews!
[+CAMERA:<id>] Camera sensor ready - Say cheese!
[+RADAR:<id>] Radar sensor ready - I'm a Radio star!
[-SENSOR:<id>] Sensor destroyed - No further data from me!
[+VEHICLE:<id>] Created at (<x>, <y>), heading <direction> at <speed> units/tick
[-CAR:<id>] Being scrapped...
[-BIKE:<id>] Being locked away...
[+PARKED:<id>] Parked at (<x>, <y>)
[-PARKED:<id>] I'm being towed away!
[+LIGHT:<id>] Initialized at (<x>, <y>) to <color>
[-LIGHT:<id>] Turning off
[+CAR:<id>] Initialized at (<x>, <y>) facing <direction> - No driver's license required!
[-CAR:<id>] Our journey is complete!
[+NAV:<id>] Hello, I'll be your GPS today
[-NAV:<id>] You've arrived! Shutting down...
```