

Επιλογή ενεργειών (action selection)

"The more we can find out about how our brains evolved from those of simpler animals, the easier the task will be."

Marvin Minsky

- [Σκοπός της εργασίας](#)
 - [Περιγραφή του συστήματος](#)
 - [Υλοποίηση της πλατφόρμας](#)
 - [Οδηγίες](#)
 - [Δείτε ακόμη ...](#)
-

Σκοπός της εργασίας

Σκοπός της εργασίας είναι η υλοποίηση ενός μοντέλου τεχνητού ζώου (artificial animal ή animat) που να πετυχαίνει να ικανοποιεί όσο το δυνατόν καλύτερα τις ανάγκες της επιβίωσής του. Οι ανάγκες αυτές είναι συνεχείς, π.χ. αν το animat τραφεί και ικανοποιηθεί προσωρινά την αντίστοιχη ανάγκη, πάλι μετά από κάποιο χρονικό διάστημα --που εξαρτάται από κάποιο ρυθμό απώλειας ενέργειας θα επανέλθει στην ίδια κατάσταση ανάγκης. Επίσης οι ανάγκες αυτές είναι αλληλοσυνδεδεμένες μεταξύ τους καθώς και με διάφορες περιβαλλοντικές συνθήκες, π.χ. η ανάγκη ενός animat για παιχνίδι είναι μεγαλύτερη όταν οι άλλες ανάγκες είναι σχετικά ικανοποιημένες, ενώ η ανάγκη ενός animat για τροφή είναι μικρότερη όταν η θερμοκρασία περιβάλλοντος είναι μεγαλύτερη. Δίνεται μία πλατφόρμα παιχνιδιού η οποία περιλαμβάνει ήδη έναν αριθμό animats που υλοποιούν διάφορα [αφελή μοντέλα συμπεριφοράς](#) και επιτρέπει τον πειραματισμό με νέα μοντέλα.

Η επίδοση ενός animat εξαρτάται όχι μόνο από το μοντέλο συμπεριφοράς του, αλλά και από το περιβάλλον μέσα στο οποίο βρίσκεται. Έτσι, ένα μοντέλο θα είναι τόσο καλύτερο όσο λιγότερο εξαρτάται από ιδιαιτερότητες του περιβάλλοντος. Αρα, ο μελετητής πρέπει να σχεδιάσει και να μελετήσει διαφορετικά περιβάλλοντα μέσα στα οποία θα δοκιμασθεί το μοντέλο του animat. Στην ιδανική περίπτωση, ένα μοντέλο είναι καλύτερο (ή ευφυέστερο) όταν πετυχαίνει άριστα σκορ ανεξαρτήτως περιβάλλοντος κόσμου.

Τα μοντέλα θα εισαχθούν τελικά σε ένα ενιαίο σύστημα και θα αξιολογηθούν σε ένα σύνολο διαφορετικών περιβαλλόντων. Εν συνεχεία θα μελετηθούν τα καλύτερα μοντέλα και θα αναλυθεί η συμπεριφορά τους.

Περιγραφή του συστήματος

Ανάγκες, κίνητρα και ερεθίσματα

Το animat έχει τέσσερεις ανάγκες:

- Τροφή
- Νερό
- Ξεκούραση
- Παιχνίδι

Σε καθεμία από αυτές τις ανάγκες αντιστοιχίζεται ένα "**ενεργειακό επίπεδο**" που έχει μία μέγιστη

τιμή (π.χ. χορτασμένος από παιχνίδι, τελείως ξεκούραστος κλπ.). Ο βαθμός ικανοποίησης μίας ανάγκης (π.χ. τροφής) μετράται σαν ένα σκορ ανάλογο του αντιστοίχου "ενεργειακού επιπέδου". Π.χ. όσον αφορά την ανάγκη της τροφής μπορεί ένα animat να έχει μέγιστο επίπεδο τροφής στο σώμα του 500, οπότε το αντίστοιχο σκορ θα είναι $\text{τρέχουσας_τιμή} / 500$.

Κάθε ανάγκη εκτός από το παιχνίδι μπορεί να ικανοποιηθεί αν το animat "καταναλώσει" κάποιο εξωτερικό αντικείμενο, π.χ. νερό. Το παιχνίδι ικανοποιείται χωρίς κατανάλωση κάποιου πράγματος, απλά το animat σταματά και παίζει. Το αντικείμενο που μπορεί να ικανοποιήσει την αντίστοιχη ανάγκη λειτουργεί σαν **ερέθισμα (stimulus)** για αυτό, ενώ το εσωτερικό ενεργειακό περιεχόμενο είναι ο **κινητήριος μοχλός (drive)**. Το animat μπορεί να αντιληφθεί την ύπαρξη κάποιου ερεθίσματος (αντικειμένου) γύρω του σε μία μικρή ακτίνα (π.χ. 3 θέσεις αρχικά στο σύστημά μας).

Όλα τα ενεργειακά επίπεδα των αναγκών μειώνονται με την πάροδο του χρόνου, δηλαδή υπάρχει μία συνεχής απώλεια που αναγκάζει το animat να αναζητεί διαρκώς την ικανοποίησή τους. Επιπλέον, οι ανάγκες του animat είναι αλληλοσυνδεδεμένες μεταξύ τους καθώς και με διάφορες περιβαλλοντικές συνθήκες. Ακολουθεί το **μοντέλο απώλειας ενεργειών**, που συμπεριλαμβάνει τις αλληλεξαρτήσεις (π.χ. όταν στο περιβάλλον υπάρχει ταυτόχρονα μεγάλη θερμοκρασία και μεγάλη υγρασία, τότε η δίψα του animat είναι μεγαλύτερη) :

```
protected void defaultComputeDrives()
{
    // Food
    if (temperatureStimulus.getIntensity() >= BigHeatPosition)
        foodLevel -= foodDecaySmallStep;
    else foodLevel -= foodDecayBigStep;
    if (foodLevel < 0) foodLevel = 0;

    // Water
    if ((temperatureStimulus.getIntensity() >= BigHeatPosition) ||
        (humidityStimulus.getIntensity() >= WetPosition))
        waterLevel -= waterDecayBigStep;
    else waterLevel -= waterDecaySmallStep;
    if (waterLevel < 0) waterLevel = 0;

    // Rest
    if ((temperatureStimulus.getIntensity() >= BigHeatPosition) &&
        (humidityStimulus.getIntensity() >= WetPosition))
        restLevel -= restDecayBigStep;
    else restLevel -= restDecaySmallStep;
    if (restLevel < 0) restLevel = 0;

    // Play (internally driven!)
    if ((foodLevel >= (MaxFoodLevel/2)) &&
        (waterLevel >= (MaxWaterLevel/2)) &&
        (restLevel >= (MaxRestLevel/2)))
        playLevel -= playDecayBigStep;
    else playLevel -= playDecaySmallStep;
    if (playLevel < 0) playLevel = 0;
}
```

Αντίληψη, απόφαση, πράξη

Σύμφωνα με τα παραπάνω, σε κάθε χρονική στιγμή το animat αναγκάζεται να "υπολογίσει" (ή ισοδύναμα να "διαβάσει") τα ερεθίσματά του και να αποφασίσει τελικά ποια ανάγκη θα ικανοποιήσει και πώς, **άμεσα** καταναλώνοντας κάποιο κατάλληλο αντικείμενο (ή παίζοντας) ή **έμμεσα** μετακινούμενος προς το ερέθισμα ή σε τελευταία ανάλυση στην τύχη. Το ζητούμενο του μοντέλου

είναι αυτός το τρίπτυχο αντίληψη-απόφαση-πράξη του animat.

Μέσα στο σύστημα ορίζονται κάποιοι τρόποι αντίληψης και πέντε αφελή μοντέλα απόφασης τα οποία μπορούν να αποτελέσουν σημείο αφετηρίας για τη μοντελοποίηση:

- **ExecuteLowestDrive**

Προσπαθεί να ικανοποιήσει την ανάγκη με τη μεγαλύτερη ενεργειακή απώλεια. Ορίζονται δύο εκδοχές σύμφωνα με τους δύο διαφορετικούς τρόπους αντίληψης που παρουσιάζονται [παρακάτω](#).

- **ExecuteBiggestStimulus**

Προσπαθεί να ικανοποιήσει την ανάγκη για την οποία έχει το ισχυρότερο ερέθισμα. Ορίζονται δύο εκδοχές σύμφωνα με τους δύο διαφορετικούς τρόπους αντίληψης που παρουσιάζονται [παρακάτω](#).

- **ExecuteAdditive**

Συνδυάζει αθροιστικά το ερέθισμα και το εσωτερικό ενεργειακό επίπεδο (επειδή η ύπαρξη και των δύο παραγόντων πρέπει να προτιμάται από την ύπαρξη μόνον του ενός). Ορίζονται δύο εκδοχές σύμφωνα με τους δύο διαφορετικούς τρόπους αντίληψης που παρουσιάζονται [παρακάτω](#).

- **ExecuteMultiplicative**

Συνδυάζει πολλαπλασιαστικά το ερέθισμα και το εσωτερικό ενεργειακό επίπεδο (όπως πριν, αλλά λαμβάνει υπόψη ότι ακόμη και ένα ισχυρό ερέθισμα μπορεί να αγνοηθεί αν η αντίστοιχη ανάγκη είναι σχεδόν ανύπαρκτη). Ορίζονται δύο εκδοχές σύμφωνα με τους δύο διαφορετικούς τρόπους αντίληψης που παρουσιάζονται [παρακάτω](#).

- **ExecuteComplex**

Εκτελεί τα **αντανακλαστικά (reflexes)** πρώτα και εν συνεχεία ακολουθεί το μοντέλο ExecuteBiggestStimulus. Τα αντανακλαστικά είναι πράξεις που εκτελούνται αυτόματα, χωρίς απόφαση. Έχει βρεθεί ότι, σε ορισμένες περιπτώσεις, μοντέλα όπου η **καταναλωτική συμπεριφορά (consummatory behavior)** ορίζεται ως αντανακλαστικό πλεονεκτούν εκείνων όπου η καταναλωτική συμπεριφορά αποτελεί και αυτή αντικείμενο απόφασης. Καταναλωτική συμπεριφορά στο σύστημά μας, ορίζεται η κατανάλωση κάποιου αντικειμένου-ερεθίσματος όταν το animat βρίσκεται πάνω του (δηλαδή, συμφέρει να πίνει κανείς νερό όταν είναι στη βρύση, παρά να ψάχνει για βρύση όταν θέλει νερό -- οπότε αρκεί η συμπεριφορά του να του εξασφαλίζει ότι θα πηγαίνει στη βρύση σε τακτά διαστήματα, όπως συμβαίνει στα άγρια ζώα). Η καταναλωτική συμπεριφορά αντιδιαστέλλεται σαν όρος στη **συμπεριφορά αναζήτησης ερεθίσματος (appetitive behavior)** που αντιστοιχεί στο κυνήγι με τη γενική έννοια.

- **Τρόποι αντίληψης**

Για κάθε θέση του κόσμου που βρίσκεται μέσα στο αντιληπτικό εύρος του animat, ορίζεται ένα ερέθισμα ανά ανάγκη, του οποίου η ένταση διαβαθμίζεται σύμφωνα με την πραγματική απόσταση από τη θέση του animat (π.χ. ένταση, ένταση/2, ένταση/3, για το ίδιο ερέθισμα σε απόσταση 0, 1, 2 από το animat). Αυτά υλοποιούνται μέσα στις μεθόδους **computeFoodStimuli()**, **computeWaterStimuli()** και **computeShadeStimuli()**. Όταν λοιπόν το μοντέλο απόφασης χρησιμοποιεί το ερέθισμα ως προς μία ανάγκη και μία συγκεκριμένη κατεύθυνση, μπορεί να το υπολογίσει είτε με τη μέγιστη ένταση που γίνεται αντιληπτή προς αυτή την κατεύθυνση (**findMaxStimulusIn**), είτε με τη συνολική αντίστοιχη ένταση (**findMaxDirectedStimulusIn**). Διαφορετικοί τρόποι αντίληψης μπορούν να ορισθούν ορίζοντας διαφορετικά τις **computeFoodStimuli()**, **computeWaterStimuli()** και **computeShadeStimuli()** ή/και ενσωματώνοντας τα αντίστοιχα ερεθίσματα με διαφορετικούς τρόπους μέσα στο μοντέλο απόφασης. Σημειώνεται τέλος ότι τόσο η αντίληψη όσο και η μετακίνηση είναι δυνατές ως προς 4 ή 8 κατευθύνσεις αντίστοιχα.

Δίνονται τέλος επτά μοντέλα **ExecuteCustom1**, **ExecuteCustom2**, **ExecuteCustom3**, **ExecuteCustom4**, **ExecuteCustom5**, **ExecuteCustom6** και **ExecuteCustom7**, από τα οποία τα πέντε πρώτα υλοποιούν τη δεύτερη εκδοχή καθενός από τα πέντε αφελή μοντέλα (μέσα στην κλάση CustomAnimat, [βλέπε οδηγίες παρακάτω](#)), ενώ τα δύο τελευταία μπορούν να χρησιμοποιηθούν για πειραματισμό και συγκριτικές δοκιμές.

Περιβάλλον

Το περιβάλλον είναι ο διδιάστατος χώρος μέσα στον οποίο υπάρχει ένας αριθμός από (παθητικά) αντικείμενα και (ενεργητικά) animats τα οποία επιπλέον κινούνται. Τα αντικείμενα αποτελούν ερεθίσματα για τις ανάγκες των animats και μπορούν να καταναλωθούν από αυτά. Π.χ. τα animats μπορούν να καταναλώσουν αντικείμενα τύπου WATER προκειμένου να ικανοποιήσουν την ανάγκη της δίψας (μεταβλητή "waterLevel"). Ένα αντικείμενο μπορεί να είναι ενός από τους εξής τρεις τύπους, καθένας από τους οποίους αντιστοιχεί σε μία από τις ανάγκες των animats, εκτός του παιχνιδιού :

- **Τροφή (FOOD)**
- **Νερό (WATER)**
- **Σκία ή δένδρο (SHADE)**

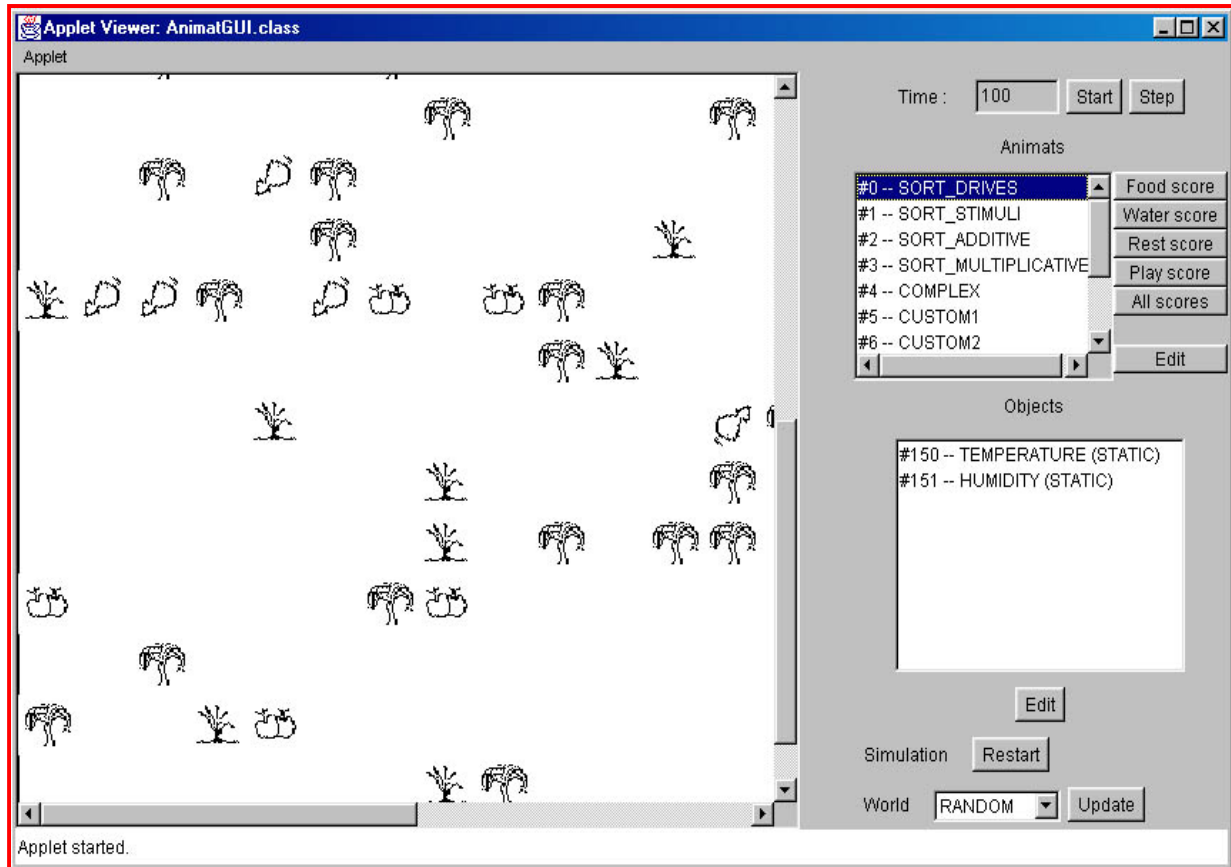
Τα αντικείμενα έχουν μία τιμή (που μειώνεται όταν ένα animat καταναλώνει ένα τμήμα τους) και μπορούν να είναι στατικά ή δυναμικά στο χρόνο. Η δυναμική αυτή μπορεί να ανήκει σε έναν από τους παρακάτω τύπους:

- **DECAYING**
Η τιμή του αντικειμένου φθίνει με το χρόνο. Π.χ. ένα οπωροφόρο δένδρο γερνά.
- **FLUCTUATING**
Η τιμή του αντικειμένου ταλαντώνεται ανάμεσα σε δύο όρια. Π.χ. ένα φυλλοβόλο δένδρο παρουσιάζει περιοδικότητα του όγκου του φυλλώματός του, άρα και της σκίας που μπορεί να προσφέρει.
- **REGENERATING**
Η τιμή του αντικειμένου ανανεώνεται σταδιακά μετά από κάθε κατανάλωσή του. Π.χ. το χόρτο ανανεώνεται.

Τέλος, ορίζονται μία σειρά "σημάτων" δηλαδή αντικειμένων που είναι όπως τα προηγούμενα ως προς τη δυναμική τους, αλλά δεν έχουν υλική υπόσταση. Τέτοια σήματα στο σύστημά μας είναι η **θερμοκρασία** και η **υγρασία**, που αποτελούν και σημαντικούς περιβαλλοντικούς παράγοντες που επηρεάζουν τη συμπεριφορά των animats (βλ. [μοντέλο απώλειας ενέργειας](#)).

Υλοποίηση της πλατφόρμας

Κάνετε "κλικ" μέσα στην εικόνα για να παίξετε



Source Java code

Εγχειρίδιο χρήσης

Η οθόνη του applet παρουσιάζει μία λευκή περιοχή στα αριστερά όπου βρίσκονται τα προσομοιωμένα ζώα (artificial animals ή animats) και αντικείμενα (τροφή, πηγές νερού και δένδρα). Ο χρήστης μπορεί να κάνει scrolling στην οθόνη της προσομοίωσης οριζόντια ή κάθετα. Με διπλό κλικ πάνω σε ένα animat ή αντικείμενο ανοίγει ένας editor. Επειδή σε μία θέση μπορεί να περιέχονται το πολύ ένα animat αλλά πάνω από ένα αντικείμενα, μία δεύτερη δυνατότητα για editing προσφέρεται με shift και κλικ: ανοίγει τότε ένας editor με δύο τμήματα, έναν επιλογέα αντικειμένων στα αριστερά και μία περιοχή στα δεξιά όπου θα ανοίξει ο editor του αντικειμένου που επιλέχθηκε στα αριστερά. Αυτός ο select-and-edit διάλογος ανοίγει αυτόματα με διπλό κλικ του χρήστη πάνω σε μία θέση που περιέχει πάνω από ένα αντικείμενα. Ο editor ενός animat είναι κατ' αρχήν do-nothing, ενώ ο editor ενός αντικειμένου επιτρέπει στο χρήστη να αλλάξει την τιμή του (quantity) και τον τύπο του (ένας από FOOD, WATER και SHADE).

Η δεξιά πλευρά περιλαμβάνει έναν αριθμό συνιστωσών ελέγχου της προσομοίωσης.

- **Το ρολόι του συστήματος.**
- **Ένα κουμπί start/stop της προσομοίωσης.**
Σε κάθε βήμα της προσομοίωσης, όλα τα animats εκτελούν με τυχαία σειρά. Εκτελούν επίσης όλα τα μη στατικά αντικείμενα (π.χ. πηγές τροφής που φθίνουν με το χρόνο).
- **Ένα κουμπί step της προσομοίωσης.**
Προχωρά την προσομοίωση κατά μία μονάδα χρόνου.
- **Μία λίστα των animats** της προσομοίωσης (τα ονόματα των animats δείχνουν τι μοντέλο συμπεριφοράς ακολουθείται, π.χ. ο 1 -- SORT_STIMULI, ακολουθεί ένα μοντέλο που προτιμά τα δυνατότερα ερεθίσματα).
- **Πέντε κουμπιά για παρουσίαση σκορ.**
Χρησιμοποιούνται για να παρουσιάσουν τα σκορ που πετυχαίνει το animat που έχει επιλεγεί

στη διπλανή λίστα: σκορ τροφής (food score), σκορ νερού (water score), σκορ ανάπαυσης (rest score), σκορ παιχνιδιού (play score) και όλα τα σκορ συγκεντρωτικά.

- **Ένα κουμπί edit για το επιλεγμένο animat.**
Ανοίγει τον editor για το επιλεγμένο animat στη διπλανή λίστα.
- **Μία λίστα των καθολικών (global) αντικειμένων** της προσομοίωσης (by default, θερμοκρασία και υγρασία).
- **Ένα κουμπί edit για το επιλεγμένο αντικείμενο.**
By default, ο editor αυτός επιτρέπει στο χρήστη να αλλάξει την τιμή (quantity) του αντικειμένου.

Τέλος, το σύστημα δίνει τις παρακάτω δυνατότητες στην κάτω δεξιά πλευρά του.

- **Restart** -- Επανεκκίνηση της προσομοίωσης. Αρχικοποιεί το ρολόι στο 0, και επανεκκινεί (reset) όλα τα animats, αλλά αφήνει άθικτο το περιβάλλον τους.
- **Update world** -- Ορίζει ένα νέο περιβάλλον για την προσομοίωση, αλλά αφήνει το ρολόι και τα animats άθικτα. Ο χρήστης μπορεί να επιλέξει έναν από τους σχηματισμούς που δίνονται για το περιβάλλον (random, custom1, custom2 or custom3) και μετά να πατήσει το κουμπί "Update".

Οι δυνατότητες αυτές της πλατφόρμας επιτρέπουν στο χρήστη να πειραματισθεί συστηματικά με διάφορα μοντέλα animats, σε προκαθορισμένους σχηματισμούς περιβάλλοντος ή με άλλους περιορισμούς της επιλογής του.

Οδηγίες

Η πλατφόρμα περιλαμβάνει τεσσάρων ειδών κλάσεις:

- **Κλάσεις interfaces.** Αυτές είναι οι κλάσεις AnimatGUI, SimulationView, GenericEditor, GenericListEditor και GenericInterface, που υλοποιούν αντίστοιχα το applet, τον canvas πάνω στον οποίο σχεδιάζεται η εικόνα της προσομοίωσης, τους δύο editors, individual και select-and-edit, που χρησιμοποιούνται για τα animats και τα αντικείμενα και ένα βοηθητικό interface.
- **Κλάσεις προσομοίωσης.** Αυτές είναι οι κλάσεις AnimatSimulation, SimObject και CustomObject, και υλοποιούν αντίστοιχα την προσομοίωση αυτή καθαυτή (π.χ. πώς κινούνται μέσα στον κόσμο τα animats χωρίς να πέφτουν το ένα πάνω στο άλλο, πώς καταναλώνουν αντικείμενα και πώς εξελίσσεται η προσομοίωση), το αντικείμενο (abstract class) και το αντικείμενο που χρησιμοποιείται σε αυτή την προσομοίωση (non-abstract class). Η τελευταία κλάση δίνεται ώστε ο χρήστης να μπορεί να τη διαμορφώσει κατάλληλα για τις ανάγκες της μελέτης του.
- **Μοντέλα animats.** Αυτές είναι οι κλάσεις Animat (abstract superclass), που υλοποιεί όλες τις βασικές λειτουργίες και δυνατότητες του animat (κίνηση, αίσθηση ερεθισμάτων, απώλεια ενέργειας) και CustomAnimat (non-abstract class). Η τελευταία κλάση δίνεται ώστε ο χρήστης να μπορεί να τη διαμορφώσει κατάλληλα για τις ανάγκες της μελέτης του.
- **Βοηθητικές κλάσεις.** Αυτές είναι οι κλάσεις Observer, History, Curve, CurveViewer, ListModifierDialog, ListModifierListener και TextAreaDialog, που υλοποιούν τη δυνατότητα συλλογής στατιστικών δεδομένων σαν σειρά τιμών που μπορεί να μεταφρασθεί σαν καμπύλη και να παρουσιασθεί σε αντίστοιχο γράφημα. Είναι επίσης η κλάση Stimulus τις οποίας τα στιγμιότυπα χρησιμοποιούνται για τα ερεθίσματα που λαμβάνουν τα animats από το περιβάλλον τους. Ο χρήστης δε χρειάζεται να ξέρει τα εσωτερικά της λειτουργίας αυτών των κλάσεων, αλλά τις χρησιμοποιεί αυτούσιες.

Ζητούνται

1. Να υλοποιηθεί ένα μοντέλο animat που να πετυχαίνει τα μέγιστα δυνατά σκορ μέσα σε διαφορετικά περιβάλλοντα. Το μοντέλο αυτό θα πρέπει να υλοποιηθεί μέσα στην κλάση CustomAnimat για πειραματισμό.

Υπόδειξη. Μελετήστε την κλάση Animat (βλέπε και [συνοπτική περιγραφή](#) παρακάτω) πριν υλοποιήσετε το μοντέλο σας. Επίσης, μελετήστε το [παράδειγμα συστηματικής μελέτης](#) και προσπαθήστε να επιτύχετε το μοντέλο σας να παρουσιάζει επιμονή και δραστηριότητα στο κενό (βλ. [παράδειγμα](#)). **ΠΡΟΣΟΧΗ!! Απαγορεύεται η χρήση μεταβλητών που να υποδηλώνουν έμμεσα ή άμεσα την απόλυτη ή σχετική θέση του animat μέσα στο περιβάλλον, π.χ. απαγορεύεται η γνώση των θέσεων των αντικειμένων ή αντίστοιχα η οδομετρική γνώση (όπως πόσες κινήσεις και πόσες στροφές έκανε το animat). Χωρίς αυτή την απαγόρευση το σύστημα δε θα ήταν βιολογικά ρεαλιστικό.**

Παραδοτέα. Η κλάση CustomAnimat και ένα κείμενο περιγραφής του μοντέλου. Αν χρειάζεται για λόγους πληρότητας ή απλότητας της παρουσίασης, μπορούν να δοθούν πολλές εκδοχές ενός μοντέλου που έχουν προκύψει και αξιολογηθεί σταδιακά.

2. Για το παραπάνω μοντέλο να δώσετε τα αποτελέσματα της συστηματικής εφαρμογής του μέσα σε διαφορετικά είδη περιβαλλόντων. Επίσης, δοκιμάστε να πειραματιστείτε με παραλλαγές του [μοντέλου απώλειας ενεργειών](#) του animat (μέθοδος `computeDrives()`), π.χ. με μη γραμμικά μοντέλα απώλειας, ή με περιοδικά μοντέλα τύπου "εναλλαγής εποχών" (όπως, εναλλάξ τα Α, Β και Γ για 10000 κύκλους το καθένα).

Υπόδειξη. Δώστε συστηματικά αποτελέσματα για διαφόρους περιβαλλοντικούς σχηματισμούς (βλέπε και [παράδειγμα συστηματικής μελέτης](#) παρακάτω). Επίσης αξιολογήστε την επίδραση παραμέτρων όπως τα αρχικά ενεργειακά επίπεδα του animat (ένα ιδανικό μοντέλο δε θα πρέπει να εξαρτάται από αυτά ή αλλιώς να μπορεί να εκμεταλλευτεί προς όφελός του τυχόν ιδιαιτερότητες των).

Παραδοτέα. Ένας ή περισσότεροι πίνακες συστηματικών αποτελεσμάτων ή/και εικόνες των σκορ και ένα κείμενο αξιολόγησής των. Αν χρειάζεται, μπορεί να οριστεί και να δικαιολογηθεί το είδος του "πειράματος" που απαιτείται για την αξιολόγηση του προτεινομένου μοντέλου, π.χ. να δοκιμασθεί μόνο σε "αραιά" περιβάλλοντα (με χαμηλή πυκνότητα αντικειμένων) ή σε περιβάλλοντα με συγκεκριμένους σχηματισμούς αντικειμένων (όπως τρίγωνα τροφής, δάση από δένδρα, "πακέτα" ερεθισμάτων του τύπου [1 τροφή--1 πηγή--1 δένδρο] κλπ.).

Περιγραφή της κλάσης Animat

Μία υποκλάση της κλάσης Animat πρέπει να υλοποιήσει τις αφηρημένες μεθόδους της Animat:

- **void reset()**
Η μέθοδος επανεκκίνησης του animat. Χρησιμοποιείται για αρχικοποίηση μεταβλητών του animat.
- **void computeDrives()**
Το [μοντέλο απώλειας ενεργειών](#) του animat. Ο χρήστης μπορεί να δοκιμάσει και τροποποιημένα μοντέλα της επιλογής του.
- **void computeFoodStimuli()**
- **void computeWaterStimuli()**
- **void computeShadeStimuli()**
[Τρόπος αντίληψης](#) του animat. Αρχικά υλοποιείται η default εκδοχή, αλλά μπορεί να τροποποιηθεί από το χρήστη για πειραματισμό.

- **void executeCustom1()**
- **void executeCustom2()**
- **void executeCustom3()**
- **void executeCustom4()**
- **void executeCustom5()**
- **void executeCustom6()**
- **void executeCustom7()**

Μοντέλα απόφασης, αρχικά άδεια, που μπορούν να χρησιμοποιηθούν για πειραματισμό. Ο χρήστης παρακινείται να "γεμίσει" ένα ή περισσότερα από αυτά με τη δική του προτεινόμενη μέθοδο απόφασης. Για το σκοπό αυτό, μπορεί κατ' αρχήν να μελετήσει τα (αφελή) [μοντέλα](#) που δίνονται για σύγκριση.

- **Panel editorPanel(GenericInterface)**
- **void editorActionPerformed(ActionEvent)**
- **void editorItemStateChanged(ItemEvent)**

Μέθοδοι που χρησιμεύουν για τους editors του animat: αρχικοποίηση editor, διαχείριση γεγονότος τύπου ActionEvent στον editor και διαχείριση γεγονότος τύπου ItemEvent στον editor.

Η κλάση Animat ορίζει τα εξής δεδομένα:

- **Point position**
- **Point heading**

Η θέση και ο προσανατολισμός του animat μέσα στο περιβάλλον.

- **final static boolean Perception_8_Connectivity**
- **final static boolean Motion_8_Connectivity**

Μεταβλητές που υποδεικνύουν αν η αντίληψη και η μετακίνηση του animat είναι δυνατές προς 8 ή 4 κατευθύνσεις (τιμές true και false, αντίστοιχα).

- **int foodSensorRange**
- **int waterSensorRange**
- **int shadeSensorRange**

Η εμβέλεια των αισθητήρων του animat για τροφή, νερό και σκία, αντίστοιχα. Οι αισθητήρες μπορούν να αντιληφθούν μία τετραγωνική περιοχή γύρω από το animat με ακτίνα την αντίστοιχη εμβέλεια.

- **double foodLevel**
- **double waterLevel**
- **double restLevel**
- **double playLevel**

Τα τρέχοντα ενεργειακά επίπεδα για καθεμία από τις ανάγκες του animat.

- **int MaxFoodLevel**
- **int MaxWaterLevel**
- **int MaxRestLevel**
- **int MaxPlayLevel**

Τα αντίστοιχα μέγιστα ενεργειακά επίπεδα.

- **float BigHeatPosition**
- **float WetPosition**

Δύο σταθερές που δείχνουν τα όρια πέρα από τα οποία το animat θεωρεί ότι ζεσταίνεται ή ότι είναι υγρό, αντίστοιχα. Χρησιμεύουν στο [μοντέλο απώλειας ενεργειών](#).

- **Vector foodStimuli**
- **Vector waterStimuli**

- **Vector shadeStimuli**
Τα ερεθίσματα που δέχεται το animat σε κάθε χρονική στιγμή της προσομοίωσης. Ενημερώνονται σε κάθε κύκλο σύμφωνα με το [μοντέλο αντίληψης](#) του animat.
- **Stimulus humidityStimulus**
- **Stimulus temperatureStimulus**
Τα ερεθίσματα θερμοκρασίας και υγρασίας που δέχεται το animat σε κάθε χρονική στιγμή της προσομοίωσης. Χρησιμοποιούν στο [μοντέλο απώλειας ενεργειών](#).
- **Stimulus playStimulus**
Το (εσωτερικό) ερέθισμα του animat για παιχνίδι. Είναι εικονικό και χρησιμοποιείται για την απόφαση.
- **int behavior**
Δείχνει ποιο συμπεριφορικό [μοντέλο απόφασης](#) ακολουθεί το animat.
- **History foodScore**
- **History waterScore**
- **History restScore**
- **History playScore**
- **History totalScore**
Τα σκορ του animat για καθεμία από τις τέσσερις ανάγκες του και το συνολικό σκορ (μέσος όρος των τεσσάρων).
- **float foodDecayBigStep**
- **float foodDecaySmallStep**
- **float waterDecayBigStep**
- **float waterDecaySmallStep**
- **float restDecayBigStep**
- **float restDecaySmallStep**
- **float playDecayBigStep**
- **float playDecaySmallStep**
Οι ρυθμοί απώλειας ενέργειας του animat σε κάθε χρονική στιγμή της προσομοίωσης. Χρησιμοποιούν στο [μοντέλο απώλειας ενεργειών](#).

Οι πιο σημαντικές μέθοδοι τις οποίες υλοποιεί η κλάση Animat είναι οι εξής:

- **public void updateScores()**
- **public double computeFoodScore()**
- **public double computeWaterScore()**
- **public double computeRestScore()**
- **public double computePlayScore()**
- **public double computeTotalScore()**
Υπολογισμός των σκορ που αντιστοιχούν στις διάφορες ανάγκες του animat και ενημέρωση των στατιστικών δεδομένων.
- **public double computeMeanFoodScore()**
- **public double computeMeanWaterScore()**
- **public double computeMeanRestScore()**
- **public double computeMeanPlayScore()**
- **public double computeMeanTotalScore()**
Υπολογισμός των μέσων σκορ που αντιστοιχούν στις διάφορες ανάγκες του animat (μέσοι όροι των μέχρι τώρα σκορ).
- **protected void visualizeFoodScore()**

- **protected void visualizeWaterScore()**
- **protected void visualizeRestScore()**
- **protected void visualizePlayScore()**
- **protected void visualizeTotalScore()**
Παρουσίαση των γραφικών παραστάσεων που αντιστοιχούν στις καμπύλες των σκορ του animat.
- **public void drawOn(Graphics, Point, Point, AnimatGUI)**
Σχεδίαση του animat στον καμβά παρουσίασης της προσομοίωσης (δηλαδή σχεδίαση μέσα στο περιβάλλον του).
- **public void execute()**
Βασική μέθοδος "εκτέλεσης" του animat. Καλεί με τη σειρά τις computeDrives(), sense(), act() και updateScores().
- **final protected void sense()**
Συλλογή αντιληπτικών δεδομένων, δηλαδή ερεθισμάτων, από το animat. Καλεί με τη σειρά τις computeFoodStimuli(), computeWaterStimuli(), computeShadeStimuli(), computeTemperatureStimulus(), και computeHumidityStimulus().
- **public void computeFoodStimuli()**
- **public void computeWaterStimuli()**
- **public void computeShadeStimuli()**
Συλλογή αντιληπτικών δεδομένων κατά ανάγκη. Ο χρήστης μπορεί να αντικαταστήσει αυτές τις μεθόδους με άλλες που υλοποιούν διαφορετικούς τρόπους αντίληψης.
- **public void computeTemperatureStimulus()**
- **public void computeHumidityStimulus()**
Συλλογή αντιληπτικών δεδομένων που αντιστοιχούν στη θερμοκρασία και την υγρασία.
- **public void act()**
Κύρια μέθοδος απόφασης του animat που καλεί εκείνο από τα [μοντέλα απόφασης](#) που αντιστοιχεί στο τρέχον συμπεριφορικό του μοντέλο .
- **public void executeAction(int, Stimulus)**
Μέθοδος εκτέλεσης της τελικής ενέργειας που έχει αποφασισθεί. Καλεί μία από τις feed(), drink(), rest() και play().
- **public void feed(Stimulus)**
- **public void drink(Stimulus)**
- **public void rest(Stimulus)**
- **public void play(Stimulus)**
Μέθοδος εκτέλεσης της τελικής ενέργειας που έχει αποφασισθεί, κατά ανάγκη. Το πρότυπο εκτέλεσης ενέργειας είναι το ακόλουθο:


```

Εάν δεν υπάρχει ερέθισμα,
τότε τυχαία κίνηση,
αλλιώς,
    εάν το animat βρίσκεται πάνω σε αντικείμενο που μπορεί
        να το ικανοποιήσει,
    τότε καταναλώνει κατάλληλη ποσότητα από αυτό,
    αλλιώς
        κινείται προς την κατεύθυνση του ερεθίσματος.

```
- **public void moveBy(Point)**

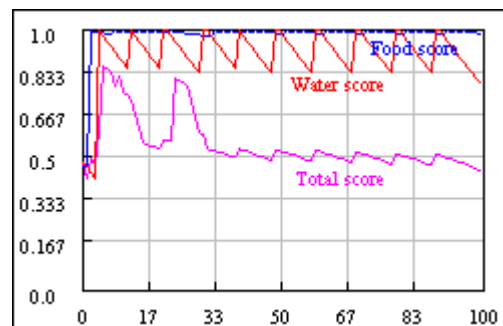
- **public void moveByTail(Point)**
- **public void moveBy(int, int)**
- **public void moveRandomly()**
Μέθοδοι κίνησης του animat. Η τυχαία κίνηση υλοποιείται σαν αλλαγή τρέχουσας κατεύθυνσης (heading) με μικρή πιθανότητα (10%) ή όταν βρεθεί εμπόδιο.
- **public void editIn(AnimatGUI)**
Ανοιγμα του editor του animat.

Παράδειγμα συστηματικής μελέτης μοντέλου

Μελετήθηκε η επίδοση των μοντέλων SORT_STIMULI (που ακολουθεί το μέγιστο ερέθισμα που αντιλαμβάνεται) και COMPLEX (που εκτελεί κατ' αρχήν τα αντανακλαστικά και εν συνεχεία κάνει ό,τι και ο SORT_STIMULI). Η προσομοίωση των δύο μοντέλων μέσα σε έναν τυχαίο κόσμο για 500 κύκλους το καθένα έδωσε τα κάτωθι αποτελέσματα:

	SORT_STIMULI	COMPLEX
Mean food score	93.3%	98.6%
Mean water score	58.6%	90.7%

Επίσης η εξέλιξη των σκορ τροφής και νερού για το animat τύπου COMPLEX δίνεται στην παρακάτω γραφική παράσταση:



Γενική ερμηνεία του μοντέλου "complex"

Τα παραπάνω αποτελέσματα είναι κατ' αρχήν τελείως αναξιόπιστα, επειδή όλα τα μοντέλα παρουσιάζουν περιοδική συμπεριφορά: πολύ σύντομα καταλήγουν σε ταλαντωτική κίνηση στο χώρο, δηλαδή το animat συμπεριφέρεται σαν αναποφάσιστο. Ωστόσο, τα αποτελέσματα δείχνουν ότι το μοντέλο COMPLEX υπερέχει σαφώς του υπο-μοντέλου του SORT_STIMULI, αφού έστω και τυχαία καταφέρνει να ικανοποιήσει τις ανάγκες του μερικές φορές. Αυτό είναι ένδειξη ότι ο ορισμός αντανακλαστικών λειτουργιών είναι επικερδής για το animat, εφόσον συνδυασθεί κατάλληλα με το μοντέλο απόφασής του.

Παρατηρήσεις

1. Το βασικό μειονέκτημα που παρουσιάζουν και τα πέντε **αφελή μοντέλα απόφασης** είναι η **αναποφασιστικότητα** της συμπεριφοράς τους: πολύ σύντομα τα μοντέλα καταλήγουν σε συμπεριφορά που παρουσιάζει ταλαντωτική κίνηση στο χώρο, δηλαδή το animat συμπεριφέρεται σαν αναποφάσιστο. Τα πραγματικά ζώα δεν παρουσιάζουν αυτό το πρόβλημα, αντίθετα η συμπεριφορά τους έχει την ιδιότητα της λεγόμενης **επιμονής (persistence)**, που σημαίνει ότι εξακολουθούν να εκτελούν την ενέργεια την κατάλληλη για την ικανοποίηση μίας ανάγκης πέρα από το σημείο στο οποίο αυτή η ανάγκη είναι η πλέον επείγουσα, ώστε να αποφευχθούν τα κόστη μεταγωγής από ανάγκη σε ανάγκη.

2. Αντίστροφα, τα πραγματικά ζώα παρουσιάζουν επιπλέον την ιδιότητα της **δραστηριότητας στο κενό (vacuum activity)**, δηλαδή δραστηριότητας που αποσκοπεί στην ικανοποίηση μίας ανάγκης παρά την απουσία του αντιστοίχου ερεθίσματος. Τα δύο αφελή μοντέλα που συνδυάζουν εξωτερικά ερεθίσματα και εσωτερικές κινητήριες ορμές, δηλαδή το ADDITIVE και το MULTIPLICATIVE, αποσκοπούν ακριβώς στην επίτευξη αυτών των δύο προδιαγραφών (επιμονή και δραστηριότητα στο κενό). Ωστόσο, όπως μπορείτε να παρατηρήσετε και μόνοι σας, όχι απλώς δε λύνεται το πρόβλημα της ταλάντωσης με αυτό τον τρόπο, αλλά τα μοντέλα δείχνουν επιπλέον **ατολμία** από την άποψη ότι είναι δυνατό να παραμένουν πάνω σε κάποιο ερέθισμα που δεν τα ικανοποιεί πλέον και να αγνοούν το περιβάλλον γύρω τους. Τελικά φαίνεται ότι απαιτείται κάποια μορφή συνδυασμού ερεθισμάτων-ορμών με κατάλληλες δυναμικές παραμέτρους.
3. Επίσης, όλα τα αφελή μοντέλα πάσχουν από **συμπεριφορική αστάθεια**, επειδή τείνουν να ζυγίζουν με διαφορετικό και μη βέλτιστο τρόπο σε κάθε κατάσταση τα υπάρχοντα ερεθίσματα, ενώ επίσης συχνά "παρασύρονται" από απομακρυσμένα ερεθίσματα περισσότερο ακόμη και από εκείνα που μπορεί να βρίσκονται στην ίδια τους τη θέση. Το πρόβλημα μπορεί να ερμηνευθεί σαν ένα πρόβλημα αναγνώρισης και σύγκρισης βραχυπροθέσμων και μακροπροθέσμων αναμενομένων οφελών.
4. Τέλος, μπορείτε να επιβεβαιώσετε ότι η συμπεριφορά του "παιχνιδιού" ρυθμίζεται πολύ δύσκολα σε κάθε περίπτωση, και το animat πετυχαίνει τα χειρότερα σκορ σε σχέση με τις άλλες ανάγκες. Η παρατήρηση αυτή αποτελεί ένδειξη ότι το "εσωτερικό" κίνητρο πρέπει να συνδυασθεί με τα υπόλοιπα με μη προφανή τρόπο.

Δείτε ακόμη ...

Ερευνητές ανά τον κόσμο που εργάζονται ή έχουν εργασθεί πάνω στο πρόβλημα της επιλογής ενεργειών

- [Mark Humphrys](#)
- [Carlos Gershenson García](#)
- [Paolo Pirjanian](#)
- [Julio Rosenblatt](#)
- [Toby Tyrrell](#)

*Τελευταία ενημέρωση 26 Μαΐου 2004.
[Στείλτε μου mail \(brensham@softlab.ece.ntua.gr\)](mailto:brensham@softlab.ece.ntua.gr)*