Μάθημα 8ο

Δημιουργία τρισδιάστατου παιχνιδιού οδήγησης 3ο μέρος

Διονύσης Κλάδης – dionisis84@yahoo.com

Προγραμματισμός Πολυμέσων - Εργαστήριο

Σε αυτό το μάθημα θα κάνουμε μία εισαγωγή στις βασικές διαδικασίες για να κάνουμε χρήση του **Havok**. Αλλά, πρώτα ας δούμε τι ακριβώς είναι το Havok.

Το Havok είναι μία μηχανή φυσικής, η οποία έχει αναπτυχθεί από την ομόνυμη ιρλανδική εταιρία Havok. Μία μηχανή φυσικής, όπως είναι το Havok, επιτρέπει την προσομοίωση σε πραγματικό χρόνο των δυνάμεων και της δυναμικής στερεών σωμάτων, και όλα αυτά σε τρεις διαστάσεις. Μας παρέχει πολλούς τύπους δυναμικών περιορισμών μεταξύ στερεών σωμάτων και έχει μία ιδιαίτερα βελτιστοποιημένη βιβλιοθήκη ανίχνευσης συγκρούσεων (collision detection). (Πηγή: WikiPedia...)

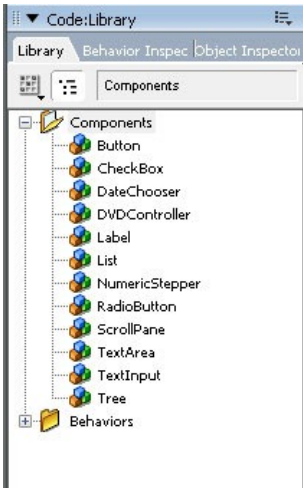
Χρησιμοποιώντας, λοιπόν, το Havok μπορούμε πολύ εύκολα να δώσουμε ρεαλισμό (ή όχι) και φυσική στα παιχνίδια μας. Το Director, όμως, δεν υποστηρίζει από την αρχή το Havok. Πρέπει να κάνουμε μία μικρή διαδικασία, έτσι ώστε να το εγκαταστήσουμε:

1. Κατεβάζουμε από το eClass τα αρχεία του Havok
    - Control.cst
    - Havok.x32
    - Setup.cst
2. Ανοίγουμε το φάκελο **Xtras** του Director (βρίσκεται εκεί που είναι εγκατεστημένο στο σκληρό δίσκο το Director, π.χ. C:\Program Files (x86)\Adobe\Adobe Director 11\Configuration\Xtras)
3. Εκεί δημιουργούμε ένα νέο φάκελο με το όνομα “Havok” και τοποθετούμε μέσα του το αρχείο Havok.x32
4. Μετά, ανοίγουμε το φάκελο **Libs** του Director (βρίσκεται εκεί που είναι εγκατεστημένο στο σκληρό δίσκο το Director, π.χ. C:\Program Files (x86)\Adobe\Adobe Director 11\Configuration\Libs)
5. Εκεί δημιουργούμε ένα νέο φάκελο με το όνομα “Havok” και τοποθετούμε μέσα του τα Setup.cst και Control.cst.

Τώρα μπορούμε να χρησιμοποιήσουμε όλη τη δύναμη του Havok στα παιχνίδια μας μέσα από το Director!

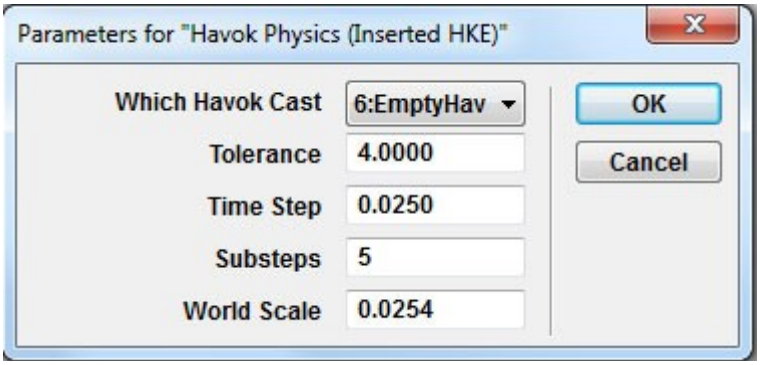
Ανοίγουμε, λοιπόν, το προηγούμενο παιχνιδάκι που δημιουργήσαμε. Το πρώτο πράγμα που πρέπει να κάνουμε είναι να εισάγουμε το αντικείμενο που μας παρέχει το Havok. Επιλέγουμε από το μενού **Insert → Media Element → Havok Physics Scene**. Ονομάζουμε το νέο cast member **EmptyHavok**.

Το επόμενο βήμα είναι να συνδέσουμε την τρισδιάστατη σκηνή που έχουμε δημιουργήσει με τη μηχανή φυσικής. Για να το κάνουμε αυτό επιλέγουμε από το Library (βρίσκεται στο κάτω μέρος της μπάρας στα δεξιά) το φάκελο **Behaviors**.



Εικόνα 1: Library

Μέσα σε αυτό το φάκελο βρίσκουμε έναν άλλο με το όνομα **Havok**. Στον φάκελο (...) **Setup** βλέπουμε διάφορες επιλογές. Αυτή που μας ενδιαφέρει αρχικά είναι η **Havok Physics (Inserted HKE)**. Το κάνουμε drag & drop στη σκηνή, **ΠΑΝΩ** στο sprite που περιέχει τον τρισδιάστατο κόσμο. Αμέσως θα εμφανιστεί το παρακάτω παραθυράκι.



Εικόνα 2: Οι παράμετροι του Havok Physics

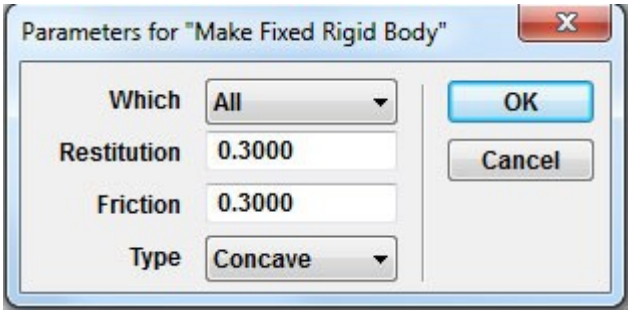
Εδώ βλέπουμε διάφορες παραμέτρους, οι οποίες καθορίζουν τη συμπεριφορά του Havok. Αναλυτικότερα, στο **Which Havok Cast** επιλέγουμε το cast member που εισάγαμε πριν λίγο. Η παράμετρος **Tolerance** καθορίζει το πόσο μεγάλη θα είναι η περιοχή γύρω από τα αντικείμενα που θα περιλάβουμε μέσα στη μηχανή φυσικής. Αυτή χρησιμοποιείται στα collisions.

Το **Timestep** είναι ο χρόνος σε ms που απαιτείται για να υπολογιστεί ξανά η φυσική. Όσο μικρότερο είναι αυτό, τόσο πιο γρήγορα γίνονται οι υπολογισμοί, απαιτώντας περισότερους υπολογιστικούς πόρους. Το **Substeps** είναι σε πόσα στάδια χωρίζεται ο κάθε κύκλος υπολογισμών. Προφανώς, όσο πιο μεγάλος είναι αυτό ο αριθμός τόσο μεγαλύτερη ακρίβεια παίρνουμε, αλλά απαιτώντας περισσότερους υπολογιστικούς πόρους.

Τέλος, το **World Scale** υποδηλώνει την κλίμακα του κόσμου. Μία μονάδα εκφράζει 1 μέτρο. Επομένως, ανάλογα με την τιμή που θα δώσουμε θα πάρουμε και την ανάλογη συμπεριφορά των αντικειμένων στο παιχνίδι. Η default τιμή εκφράζει ίντσες.

Στο δικό μας παιχνίδι αλλάζουμε το Time Step σε **0.03** και το World Scale σε **0.01** (χιλιοστά δηλαδή).

Επόμενο βήμα είναι να προσδιορίσουμε ποια αντικείμενα είναι σταθερά στερεά σώματα, έτσι ώστε το Havok να υπολογίζει κατάλληλα τις δυνάμεις. Από το library → Havok → Setup κάνουμε drag & drop ένα **Make Fixed Rigid Body** μέσα στο sprite με τον κόσμο. Αμέσως θα ανοίξει ένα παράθυρο με διάφορες ιδιότητες.



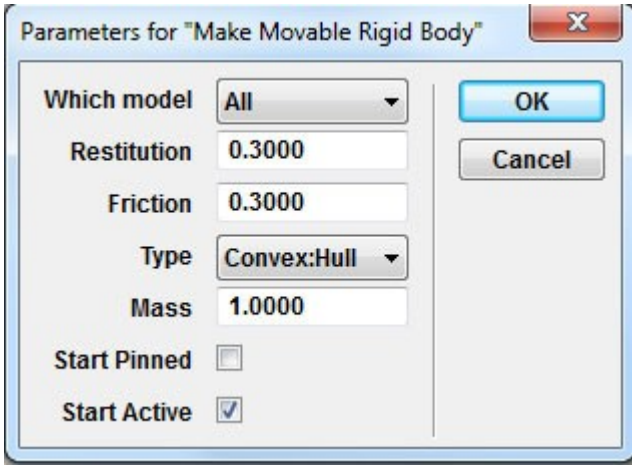
Εικόνα 3: Παράμετροι σταθερού στερεού σώματος

Το πεδίο **Which** περιλαμβάνει μία λίστα με όλα τα τρισδιάστατα αντικείμενα που υπάρχουν στον κόσμο που έχουμε δημιουργήσει. Εδώ επιλέγουμε τα αντικείμενα που θέλουμε να έχουν collisions και φυσική αλλά είναι ακίνητα. Προφανώς, μπορούμε να επιλέξουμε μόνο ένα κάθε φορά, άρα κάνουμε την ίδια διαδικασία για κάθε αντικείμενο που θέλουμε.

Η ιδιότητα **Restitution** εκφράζει την αναπήδηση που θα έχει ένα αντικείμενο που θα έρθει σε επαφή με το εν λόγω σταθερό στερεό σώμα. Η παράμετρος **Friction** εκφράζει την τριβή που θα συναντά ένα αντικείμενο που θα έρθει σε επαφή με το σταθερό στερεό σώμα.

Τέλος, το **Type** καθορίζει τον τύπο του bounding box που θα περιβάλλει το στερεό σώμα και ουσιαστικά καθορίζει την συμπεριφορά των collisions. Προσοχή, ανάλογα με το αντικείμενο που έχουμε πρέπει να επιλέξουμε και το κατάλληλο bounding box, έτσι ώστε να μη σπαταλάμε χωρίς λόγο υπολογιστική ισχύ, αλλά ούτε και να χάνουμε σε ακρίβεια.

Το επόμενο βήμα είναι να προσδιορίσουμε ποιά αντικείμενα είναι κινούμενα στερεά σώματα, και πάλι για να γίνονται σωστά οι υπολογισμοί των δυνάμενων από το Havok. Από το Library → Havok → Setup κάνουμε drag & drop ένα **Make Movable Rigid Body** μέσα στο sprite με τον κόσμο. Αμέσως θα ανοίξει ένα παράθυρο με διάφορες ιδιότητες.



Εικόνα 4: Παράμετροι κινούμενου στερεού σώματος

Το πεδίο **Which** περιλαμβάνει μία λίστα με όλα τα τρισδιάστατα αντικείμενα που υπάρχουν στον κόσμο που έχουμε δημιουργήσει. Εδώ επιλέγουμε τα αντικείμενα που θέλουμε να έχουν collisions και φυσική αλλά είναι ακίνητα. Προφανώς, μπορούμε να επιλέξουμε μόνο ένα κάθε φορά, άρα κάνουμε την ίδια διαδικασία για κάθε αντικείμενο που θέλουμε.

Η ιδιότητα **Restitution** εκφράζει την αναπήδηση που θα έχει ένα αντικείμενο που θα έρθει σε επαφή με το εν λόγω σταθερό στερεό σώμα. Η παράμετρος **Friction** εκφράζει την τριβή που θα συναντά ένα αντικείμενο που θα έρθει σε επαφή με το σταθερό στερεό σώμα.

Το **Type** καθορίζει τον τύπο του bounding box που θα περιβάλλει το στερεό σώμα και ουσιαστικά καθορίζει την συμπεριφορά των collisions. Προσοχή, ανάλογα με το αντικείμενο που έχουμε πρέπει να επιλέξουμε και το κατάλληλο bounding box, έτσι ώστε να μη σπαταλάμε χωρίς λόγο υπολογιστική ισχύ, αλλά ούτε και να χάνουμε σε ακρίβεια.

Η επόμενη ιδιότητα είναι η **Mass**. Αυτή προσδιορίζει τη μάζα (δηλαδή το βάρος) του στερού σώματος (σε κιλά ή ότι άλλο ορίζει το World Scale). Μεγαλύτερη τιμή μας δίνει πιο βαρύ αντικείμενο, το οποίο θέλει και μεγαλύτερη δύναμη για να μετακινηθεί.

Το **Start Pinned** ορίζει εάν το αντικείμενο θα είναι καρφωμένο στη θέση του όταν ξεκινήσει το παιχνίδι. Το **Start Active** ορίζει εάν το αντικείμενο θα μπορεί να δεχθεί δυνάμεις κλπ όταν ξεκινήσει το παιχνίδι.

Τώρα έχουμε τελειώσει με την αρχικοποίηση του συστήματος φυσικής, το παιχνίδι ήδη συμπεριφέρεται με βάση τους νόμους της φυσικής και το μόνο που μένει είναι να αλλάξουμε κατάλληλα τον κώδικα, έτσι ώστε να εκμεταλλευτούμε τις δυνατότητες του Havok.

Κώδικας για το movie script

```
global speedvalue
global trans
global ghavok
```

```
on startMovie
  member("rally4w").resetWorld()
  speedvalue=100

  member("rally4w").model("board").addChild(member("rally4w").model("front_t"))
  member("rally4w").model("board").addChild(member("rally4w").model("FRwheel"))
  member("rally4w").model("board").addChild(member("rally4w").model("FLwheel"))
  member("rally4w").model("board").addChild(member("rally4w").model("RRwheel"))
  member("rally4w").model("board").addChild(member("rally4w").model("RLwheel"))
  member("rally4w").model("board").addChild(member("rally4w").model("rear_t"))
  member("rally4w").model("board").addChild(member("rally4w").camera("DefaultView"))
End
```

Εδώ, πρακτικά, κάνουμε reset τον τρισδιάστατο κόσμο όταν ξεκινήσει το παιχνίδι και κάνουμε τις αρχικές ρυθμίσεις στα μοντέλα μας (όπως και στο προηγούμενο μάθημα).

Κώδικας για το frame script

```
global speedvalue
global rightturn
global leftturn
global a

global ghavok
global trans
global rbcar

on exitFrame me
  go to the frame
end

on enterFrame me
  pGoingForward  = keyPressed(126)
  pGoingBackward = keyPressed(125)
  pGoingRight    = keyPressed(124)
  pGoingLeft     = keyPressed(123)
  pSpeedUp=(keyPressed("a") or keyPressed("A"))
  pSpeedDown=(keyPressed("z") or keyPressed("Z"))
  if pGoingForward=true and speedvalue>0 then forwardmove
  if pGoingBackward=true and speedvalue>0 then backwardmove
  if pGoingRight=true and speedvalue>0 then rightmove
  if pGoingLeft=true and speedvalue>0 then leftmove
  if pSpeedUp=true then speedup
  if pSpeedDown=true then speeddown

  a=0.01

  rbcar=ghavok.rigidbody("board")
  rbcar.friction=0.5

  rbterrain=ghavok.rigidbody("ground")
  rbterrain.friction=0.9
end

on speedup
  if speedvalue<2000 then speedvalue=speedvalue+50
end

on speeddown
  speedvalue=speedvalue-50
  if speedvalue<0 then speedvalue=0
```

```
end

on forwardmove
  trans = member("rally4w").model("board").transform.duplicate()
  trans.position = Vector(0,0,0)
  trans.scale = Vector(1,1,1)
  currentFwd = trans * vector(0,speedvalue,0)

  rbcar.applyimpulse(currentFwd)
end

on leftmove
  trans = member("rally4w").model("board").transform.duplicate()
  trans.position = Vector(0,0,0)
  trans.scale = Vector(1,1,1)
  currentFwd = trans * vector(0,1,0)
  currentAxis = trans * vector(-1,0,0)
  currentleft = a*(currentFwd.cross(currentAxis))

  rbcar.applyangularimpulse(currentleft)
end

on rightmove
  trans = member("rally4w").model("board").transform.duplicate()
  trans.position = Vector(0,0,0)
  trans.scale = Vector(1,1,1)
  currentFwd = trans * vector(0,1,0)
  currentAxis = trans * vector(1,0,0)
  currentleft = currentFwd.cross(currentAxis)
  currentright=1*a*currentleft

  rbcar.applyangularimpulse(currentRight)
end


on backwardmove
  trans = member("rally4w").model("board").transform.duplicate()
  trans.position = Vector(0,0,0)
  trans.scale = Vector(1,1,1)
  currentFwd = trans * vector(0,speedvalue,0)

  rbcar.applyimpulse(-1*currentFwd)
end
```

Και εδώ, η λογική του κώδικα είναι ίδια με τις προηγούμενες φορές. Το μόνο που αλλάζει είναι ο τρόπος με τον οποίο γίνεται η μετακίνηση και η περιστροφή του οχήματος. Αυτό γίνεται χρησιμοποιώντας εντολές που μας παρέχει το Havok, όπως είναι για παράδειγμα η applyImpulse.

Περισσότερες πληροφορίες, όπως και εντολές/functions, για το Havok μπορείτε να βρείτε στο manual που υπάρχει στα έγγραφα στο eClass.

Άσκηση για το σπίτι

Το μόνο που έχετε να κάνετε σαν άσκηση στο σπίτι είναι να προσαρμόσετε το δικό σας παιχνίδι (αυτό με το όχημα/ανθρωπάκι/ούφο/τριλοβίτη) έτσι ώστε να κάνει χρήση της μηχανής φυσικής Havok. Θα πρέπει, δηλαδή, το όχημα σας να μετακινείται μέσω του Havok, όπως επίσης

τα collisions, οι δυνάμεις, η βαρύτητα κλπ να προέρχονται από το Havok.

Στο eClass θα ανεβάσετε ένα rar/zip αρχείο, το οποίο θα περιέχει όλα τα απαραίτητα για τη σωστή λειτουργία του παιχνιδιού αρχεία (εικόνες, ήχους ή οτιδήποτε άλλο).