ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

# Επικοινωνία Ανθρώπου – Μηχανής με έμφαση στην Εμψύχωση Χαρακτήρων

***Διπλωματική Εργασία***

Λιάπης Αντώνιος

**Επιβλέπων:** Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π

Αθήνα, Ιανουάριος 2007

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

# Επικοινωνία Ανθρώπου – Μηχανής με έμφαση στην Εμψύχωση Χαρακτήρων

***Διπλωματική Εργασία***

Λιάπης Αντώνιος

**Επιβλέπων :** Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ..... Ιανουαρίου 2007

| ..... | ..... | ..... |
|---|---|---|
| Στέφανος Κόλλιας | Ανδρέας Σταφυλοπάτης | Παναγιώτης Τσανάκας |
| *Καθηγητής Ε.Μ.Π.* | *Καθηγητής Ε.Μ.Π.* | *Καθηγητής Ε.Μ.Π.* |

Αθήνα, Ιανουάριος 2007

# Περίληψη

Η εφαρμογή που αναπτύχθηκε και θα περιγραφεί σ' αυτήν την αναφορά αφορά την περιήγηση μέσα σε ένα πεπερασμένο τρισδιάστατο κόσμο που αναπαριστά ένα μουσείο, και είναι γραμμένη σε γλώσσα TorqueScript. Για την ολοκλήρωση της εφαρμογής απαιτήθηκε ο σχεδιασμός των τρισδιάστατων αντικειμένων και των κτιριακών δομών που χρησιμοποιούνται από τον εικονικό κόσμο, η κατασκευή του γραφικού περιβάλλοντος, ο σχεδιασμός της τεχνητής νοημοσύνης του ξεναγού αλλά και ο ορισμός της συμπεριφοράς των αλληλεπιδραστικών αντικειμένων της εφαρμογής. Οι παραπάνω διεργασίες περιγράφονται αναλυτικά στα κεφάλαια που ακολουθούν. Τα θέματα που επεξεργάζεται κάθε κεφάλαιο αναφέρονται συνοπτικά στη συνέχεια:

- **Το 1ο κεφάλαιο** είναι μια εισαγωγή για την μηχανή Torque. Περιγράφει εν συντομία τα χαρακτηριστικά του, τα προτερήματα και τα μειονεκτήματά της.
- **Το 2ο κεφάλαιο** περιγράφει τα προγράμματα που χρησιμοποιήθηκαν για την κατασκευή της εφαρμογής.
- **Το 3ο κεφάλαιο** περιγράφει τη δομή της εφαρμογής με συνοπτικές περιγραφές των φακέλων που απαρτίζουν. Επίσης περιγράφονται τα πιο σημαντικά αρχεία στον φάκελο common.
- **Το 4ο κεφάλαιο** περιγράφει τους τρόπους αλληλεπίδρασης του χρήστη με την εφαρμογή. Αναλύεται η νοοτροπία πελάτη/εξυπηρετητή και πώς υλοποιείται στο Torque, οι εισαγωγικές οθόνες, το γραφικό περιβάλλον της εφαρμογής και οι τρόποι επικοινωνίας του πελάτη με τον εξυπηρετητή.
- **Το 5ο κεφάλαιο** περιγράφει σε συντομία τον ψηφιακό κόσμο, αναλύει τους διαφορετικούς τύπους αντικειμένων που χρησιμοποιήθηκαν, εξηγεί της διαφορές μεταξύ object και DataBlock και παρέχει μια έποψη της εφαρμογής.
- **Το 6ο κεφάλαιο** περιγράφει τα διάφορα αλληλεπιδραστικά αντικείμενα της εφαρμογής. Γίνεται εκτενής περιγραφή των συναρτήσεων που ορίζουν τη συμπεριφορά του κάθε αντικειμένου, συνοδευόμενη από τον κώδικα του αρχείου που το περιγράφει.
- **Το 7ο κεφάλαιο** περιγράφει την τεχνητή νοημοσύνη του ξεναγού της εφαρμογής. Αναλύονται οι μέθοδοι με τις οποίες ανιχνεύει τη θέση του χρήστη, υλοποιεί αλλαγές στη συμπεριφορά του, ακολουθεί και μιλά στο χρήστη, και όλες τις άλλες ενέργειες που απαιτούνται για μια ρεαλιστική συμπεριφορά.

# Abstract

The application that has been developed and will be described in this essay deals with the walkthrough in a finite 3D world representing a museum, and is written in the programming language TorqueScript. For the completion of the application a number of tasks were required, including the design of the 3D objects and buildings used in the virtual world, the construction of the interfaces, the definition of the artificial intelligence of the NPC-guide and the description of the behavior of the interactive objects. All the above are analyzed in the chapters to follow. The issues every chapter deals with are presented in short below:

- **The 1st chapter** is an introduction to the Torque Game Engine. It provides brief insight to its characteristics, its strengths and its shortcomings.
- **The 2nd chapter** describes the programs used to construct this application.
- **The 3rd chapter** describes the structure of the application with concise explanations of the folders that assemble it. The most important files in the common folder are also presented.
- **The 4th chapter** describes the methods of interaction between user and machine. The client/server concept is presented as well as the way it is applied in Torque, the splashscreens, the interface of the application and the methods of messaging between client and server.
- **The 5th chapter** describes the virtual world in short, analyzes the different object types used therein, explains the differences between object and DataBlock and offers an outlook of the application's mission.
- **The 6th chapter** describes the various interactive objects of the application. A thorough analysis of the functions that define the behavior of each object takes place, followed by the code of the file that describes it.
- **The 7th chapter** describes the artificial intelligence of the walkthrough's guide. It illustrates the functions that detect the avatar's position, that incur changes in its behavior, that follow and talk to the user's avatar, as well as all the other actions required for a realistic behavior.

# Λέξεις-Κλειδιά

- **Avatar:** Το μοντέλο μέσω του οποίου ο χρήστης μπορεί να περιηγείται μέσα στην εφαρμογή. Ως επί το πλείστον, η κίνηση του avatar γίνεται από τον χρήστη.
- **Bot:** Το μοντέλο του ξεναγού το οποίο κινείται και δρα βάσει κώδικα ο οποίος διαχειρίζεται την τεχνητή του νοημοσύνη.
- **NPC (Non-Player Character):** Κάθε χαρακτήρας που δεν ελέγχεται από κάποιον χρήστη αλλά κινείται αυτοβούλως βάσει του κώδικα τεχνητής νοημοσύνης. Στην εφαρμογή ο μόνος NPC είναι ο ξεναγός του μουσείου.
- **ID:** Ο αριθμός τον οποίο δίνει η εφαρμογή σε κάθε αντικείμενο. Συνήθως τα διαφορετικά αντικείμενα αναγνωρίζονται μέσω του ID τους.
- **Handle:** Κατά μία έννοια το handle είναι όμοιο με το ID. Τα πάντα στην εφαρμογή έχουν ένα handle μέσω του οποίου μπορούν να αναγνωριστούν.
- **Inventory:** Όταν ο χρήστης παίρνει ένα αντικείμενο για να το χρησιμοποιήσει σε ένα άλλο σημείο (ή αντικείμενο) της εφαρμογής τότε το αντικείμενο αυτό τοποθετείται στο inventory του.
- **Interface:** Το γραφικό περιβάλλον.
- **GUI (Graphic User Interface):** Το γραφικό περιβάλλον.
- **Interactive:** Αλληλεπιδραστικός, δηλαδή ένα αντικείμενο το οποίο αντιδρά σε ενεργοποίηση από τον χρήστη.
- **Animation:** Πολλά μοντέλα στην εφαρμογή κινούνται μέσα από animation sequences, δηλαδή σειρές από προδιαγεγραμμένες
- **Torque:** Η μηχανή η οποία χρησιμοποιείται για να τρέχει την παρούσα εφαρμογή. Δημιουργία της GarageGames.
- **TorqueScript:** Το κομμάτι κώδικα της μηχανής Torque. Είναι η γλώσσα που χρησιμοποιείται σε όλη την εφαρμογή.
- **DataBlock:** Ειδικός τύπος αντικειμένου που περιέχει ένα σύνολο χαρακτηριστικών που χρησιμοποιούνται για να περιγράψουν τις ιδιότητες ενός άλλου αντικειμένου.

# Περιεχόμενα

# ΚΕΦΑΛΑΙΟ 1:
# Λίγα λόγια για το Torque

## 1.1: Ιστορία

Το Torque Game Engine, ή TGE, είναι μια τροποποιημένη έκδοση μιας τρισδιάστατης μηχανής παιχνιδιών υπολογιστών που αναπτύχθηκε αρχικά από την Dynamix για το παιχνίδι Tribes 2. Το Torque είναι από τότε διαθέσιμο με την άδεια της GarageGames στους ανεξάρτητους και επαγγελματίες δημιουργούς παιχνιδιών.

Αμέσως μετά από την εμφάνιση του Tribes 2, πολλά μέλη της ομάδας Dynamix έφυγαν για να δημιουργήσουν την επιχείρησή GarageGames και αγόρασαν τη μηχανή του Tribes 2. Μετά από την εκτενή τροποποίηση δημιουργήθηκε το Torque Game Engine.

## 1.2: Χαρακτηριστικά

Εκτός από μηχανή τρισδιάστατων γραφικών, το Torque παρέχει συμπαγή κώδικα δικτύου, scripting, διαμόρφωσή του κόσμου σε πραγματικό χρόνο και δημιουργία γραφικού περιβάλλοντος. Ο κώδικας μπορεί να συμπιλιστεί σε Windows, Macintosh ή Linux.
Το Torque περιέχει μια μηχανή κατασκευής εδάφους (terrain) που δημιουργεί αυτόματα Levels-of-Detail (LODs) έτσι ώστε αναπαριστά τα ελάχιστα απαραίτητα πολύγωνα οποιαδήποτε στιγμή. Το έδαφος «φωτίζεται» αυτόματα και οι τα textures του εδάφους μπορούν να συνδυαστούν μεταξύ τους.
Το Torque υποστηρίζει τη φόρτωση των τρισδιάστατων μοντέλων σε μορφή DTS και DIF.

- Τα μοντέλα DTS μπορούν να έχουν animation χρησιμοποιώντας είτε bone structure είτε morphing. Είναι επίσης δυνατό να συνδυαστούν πολλαπλά bone animations μαζί στο παιχνίδι για τη δημιουργία animated χαρακτήρων.
- Τα μοντέλα DIF έχουν προϋπολογισμένο φωτισμό και χρησιμεύουν για την αναπαράσταση μεγάλων μοντέλων όπως κτήρια.

Το Torque υποστηρίζει παιχνίδια μέσω διαδικτύου ή LAN χρησιμοποιώντας την παραδοσιακή αρχιτεκτονική πελάτη/εξυπηρετητή. Τα αντικείμενα του εξυπηρετητή αντιγράφονται στους πελάτες και ανανεώνονται περιοδικά ή σε κάποια σημεία του παιχνιδιού.

## 1.3: Δυνατά σημεία

Μια εκτενής κοινότητα ανάπτυξης ανεξάρτητων παιχνιδιών έχει συσπειρωθεί γύρω από TGE, εν μέρει και λόγω της χαμηλής τιμής του. Ενώ η ποιότητα του Torque μπορεί να αντιστοιχηθεί ή να ξεπεραστεί από άλλες ελεύθερες, χαμηλού κόστους, ή open-source μηχανές, πολλοί χρήστες θεωρούν ότι το TGE προσφέρει μια πλήρη μηχανή παιχνιδιών, κάτι που άλλες μηχανές του ίδιου κόστους δεν μπορούν να προσφέρουν.

### *1.3.1: Networking*

Το πιο πολυδιαφημισμένο χαρακτηριστικό του Torque είναι η δυνατότητά του να συνδέεται με άλλα προγράμματα μέσω διαδικτύου. Θεωρητικά έχει εξαιρετικά χαμηλό latency και είναι σε θέση να τρέξει online παιχνίδια χωρίς σημαντικό lag.

### *1.3.2: Κοινότητα*

Η κοινότητα των χρηστών του TGE συνεισφέρει σημαντικά στη διαμόρφωση του προγράμματος. Τα διάφορα plugins των χρηστών βρίσκονται στην ιστοσελίδα της GarageGames και μπορεί ο καθένας να τα χρησιμοποιήσει. Μεγάλο μέρος της επιτυχίας του Torque βασίζεται στη συμμετοχή των ανεξάρτητων χρηστών σε αυτό με τα mods τους.

## 1.4: Μειονεκτήματα

Μερικοί χρήστες του Torque θεωρούν ότι δεν είναι ένα καλοφτιαγμένο προϊόν. Πολλοί άνθρωποι έχουν εκφράσει την απέχθεια τους για το TGE. Μεταξύ άλλων λόγων, αναφέρονται το κακό format των τρισδιάστατων αντικειμένων, το κακό documentation, παραπλανητικό marketing, κατώτερη ηχητική υποστήριξη σε σύγκριση με το tribes 2, έλλειψη map editors, ξεπερασμένα γραφικά, και τον ανοργάνωτο, με λάθη κώδικα.

### *1.4.1: Documentation*

Όσοι κριτικάρουν το Torque συχνά αναφέρουν το documentation ως μια από τις μεγάλες αδυναμίες του. Ενώ το μέγεθος του documentation είναι ικανοποιητικό, η ποιότητά του είναι κακή και οι πληροφορίες που περιέχει ελάχιστες. Καθώς το TGE έχει αρκετές ιδιαιτερότητες που δεν μπορούν να κατανοηθούν με απλή παρατήρηση του κώδικα. Η GarageGames έχει κατέβαλε προσπάθειες να μετριαστεί η κριτική με τη δημιουργία του Torque Development Network που είναι ένα κλειστό wiki για όσους έχουν αγοράσει τη μηχανή.

### *1.4.1: Ήχος*

Η υποστήριξη που έχει το Torque στον τομέα του ήχου είναι σημαντικά λιγότερο από την αντίστοιχη στο tribes 2, απ' όπου ξεκίνησε. Για να διατηρήσει η GarageGames τη χαμηλή τιμή για το TGE, έπρεπε να χρησιμοποιήσει την μόνη cross-platform ηχητική βιβλιοθήκη: το OpenAL. Όμως η ποιότητα του OpenAL και η υποστήριξή του είναι σημαντικά χαμηλότερες από άλλες ηχητικές βιβλιοθήκες.

# ΚΕΦΑΛΑΙΟ 2:
# Εργαλεία που χρησιμοποιήθηκαν

Το πρόγραμμα που είναι υπεύθυνο για την εκτέλεση της εφαρμογής, που επιτρέπει την πλοήγηση στον εικονικό κόσμο του μουσείου είναι το Torque Game Engine. Το πρόγραμμα αυτό αναλαμβάνει την μεταγλώττιση και φόρτωση του κώδικα της εφαρμογής και όλων των απαραίτητων αρχείων.

Πέραν του Torque Game Engine χρησιμοποιήθηκε μια σειρά άλλων προγραμμάτων για τον σχεδιασμό των αντικειμένων που απαρτίζουν τον εικονικό κόσμο. Έτσι χρησιμοποιήθηκε το 3DStudio MAX για την κατασκευή των κτιριακών δομών και των αντικειμένων και το Ulead PhotoImpact για την δημιουργία και επεξεργασία των textures αυτών των αντικειμένων. Επιπλέον χρησιμοποιήθηκε το Torque ShowTool Pro για την δοκιμή της κίνησης των χαρακτήρων πριν την εισαγωγή τους στον εικονικό κόσμο.

## 2.1: Torque Game Engine

Όπως στην περίπτωση των περισσότερων Game Engines το Torque περιλαμβάνει τα παρακάτω τμήματα (modules): τμήμα τρισδιάστατων γραφικών, τμήμα ήχου, τμήμα τεχνητής νοημοσύνης, τμήμα ανίχνευσης σύγκρουσης, τμήμα εισόδου/εξόδου, τμήμα βάσης δεδομένων, τμήμα δικτύου και τμήμα γραφικού περιβάλλοντος. Το Torque υποστηρίζει κώδικα σε μορφή script. Το **TorqueScript** (όπως αποκαλείται) ακολουθεί το αντικειμενοστρεφές μοντέλο και υποστηρίζει τον ορισμό κλάσεων και DataBlocks. Κάθε αντικείμενο που κατασκευάζεται με χρήση της δεσμευμένης λέξης new ανήκει σε μία κλάση. Τα χαρακτηριστικά του αντικειμένου καθορίζονται από τις τιμές των πεδίων της κλάσης και η συμπεριφορά του από της μεθόδους που υλοποιεί. Ορισμένα αντικείμενα χρησιμοποιούν DataBlocks προκειμένου να αποκτήσουν επιπρόσθετες ιδιότητες ή να χρησιμοποιήσουν συναρτήσεις κοινές με άλλα αντικείμενα. Το TorqueScript υποστηρίζει επιπλέον την κληρονομικότητα και τον πολυμορφισμό. Το συντακτικό της γλώσσας μοιάζει με αυτό της C++ με κύρια διαφορά ότι στο TorqueScript δεν απαιτούνται δηλώσεις τύπων των μεταβλητών που χρησιμοποιούνται. Η script language κάνει διάκριση μεταξύ τοπικών και παγκόσμιων μεταβλητών (οι τοπικές μεταβλητές έχουν το πρόθεμα % ενώ οι παγκόσμιες το πρόθεμα $) και χρησιμοποιεί τις βασικότερες δομές ελέγχου ροής του προγράμματος και τους αριθμητικούς και λογικούς τελεστές, που χρησιμοποιεί και η C++. Επίσης υποστηρίζει δυναμικό φόρτωμα πακέτων του κώδικα όταν αυτό απαιτείται από την τρέχουσα εφαρμογή. Ο κώδικας του Torque περιλαμβάνει τον ορισμό περίπου 420 έτοιμων συναρτήσεων, 680 μεθόδων και 200 callbacks. Εκτός από τις προκαθορισμένες συναρτήσεις και μεθόδους του Torque δίνεται η δυνατότητα στον χρήστη να ορίσει και να χρησιμοποιήσει τις δικές του συναρτήσεις και μεθόδους.

Μέσω κώδικα γραμμένου σε TorqueScript μπορούν να ελεγχθούν όλες οι λειτουργίες των εφαρμογών που εκτελούνται στο Torque Game Engine, όπως η δημιουργία και καταστροφή αντικειμένων, ανίχνευση εισόδου και ανταπόκριση σε αυτή, ανίχνευση σύγκρουσης μεταξύ αντικειμένων, δημιουργία συνδέσεων σε εφαρμογές που ακολουθούν την αρχιτεκτονική πελάτη/εξυπηρετητή, κατασκευή γραφικού περιβάλλοντος, καθορισμός θέσης, περιστροφής και κλίμακας των αντικειμένων που εμφανίζονται στο περιβάλλον της εφαρμογής, καθορισμός animation και ταχύτητας κίνησης χαρακτήρων, χειρισμός ήχου, οργάνωση χρονοδιαγράμματος και ορισμός ακολουθιών διαδοχικών ενεργειών.
Εκτός από το TorqueScript, το Torque Game Engine έχει κάποιες στοιχειώδεις WYSIWYG εφαρμογές για την κατασκευή του ψηφιακού κόσμου και του γραφικού περιβάλλοντος. Αυτές οι εφαρμογές είναι ο **world editor** και ο **GUI editor** αντίστοιχα.

Ο **world editor** χρησιμοποιείται για τον σχεδιασμό του περιβάλλοντος της εφαρμογής και περιλαμβάνει τον **terrain editor** για την διαμόρφωση του «εδάφους» της εφαρμογής. Η επιλογή του **world inspector** εμφανίζει μια δενδρική δομή στο δεξί μέρος της οθόνης, όπου εμφανίζονται όλα τα αντικείμενα που αποτελούν το περιβάλλον του εικονικού κόσμου συνοδευόμενα από το ID τους. Με επιλογή οποιουδήποτε αντικειμένου της δενδρικής δομής εμφανίζονται πληροφορίες που αφορούν την θέση του, την περιστροφή του, το μέγεθός του και μία σειρά άλλων χαρακτηριστικών ιδιοτήτων, ανάλογα με το είδος του αντικειμένου. Τα αντικείμενα του περιβάλλοντος μπορούν να ομαδοποιηθούν με χρήση αντικειμένων SimGroup που λειτουργούν σαν φάκελοι και συμβάλλουν στην καλύτερη οργάνωση της δενδρικής δομής του εικονικού κόσμου. Τα αντικείμενα που απαρτίζουν το περιβάλλον του εικονικού κόσμου κατασκευάζονται μέσω του **world creator**. Ο world creator χωρίζει τα αντικείμενα αυτά σε 4 κατηγορίες: αντικείμενα τύπου interiors, shapes, static shapes και mission objects. Μετά την κατασκευή ενός αντικειμένου δίνεται η δυνατότητα στον χρήστη να ορίσει την θέση που θα καταλαμβάνει στον εικονικό κόσμο, το όνομα του και τα υπόλοιπα χαρακτηριστικά του. Η επιλογή του **mission area editor** ανοίγει ένα παράθυρο που επιτρέπει μια γενική επισκόπηση του χώρου όπου μπορούν να τοποθετηθούν τα κτίρια και τα υπόλοιπα αντικείμενα του εικονικού κόσμου. Με την βοήθεια του mission area editor οριοθετείται ο χώρος που θα καταλαμβάνει το περιβάλλον του εικονικού κόσμου και καθορίζεται το μέγεθος του. Ο world editor παρέχει επιπλέον επιλογές που έχουν να κάνουν με την θέση, την λήψη και την ταχύτητα κίνησης της κάμερας και περιλαμβάνονται στο μενού camera. Μετά την τοποθέτηση ή την αλλαγή της θέσης οποιουδήποτε αντικειμένου είναι απαραίτητη η επιλογή του relight scene που υπολογίζει τις σκιές που δημιουργεί το νέο αντικείμενο στο περιβάλλον του.

Ο **terrain editor**, όπως προείπαμε, χρησιμοποιείται για την διαμόρφωση του «εδάφους» της εφαρμογής. Κατά την επιλογή του εμφανίζεται το εργαλείο *brush* που επιτρέπει την διαμόρφωση της τοπολογίας της εφαρμογής με δημιουργία υψωμάτων στο έδαφος. Το μέγεθος του εργαλείου brush, η ισχύς του και το σχήμα του ρυθμίζονται μέσω των επιλογών του μενού brush. Το εργαλείο **terrain terraform editor** χρησιμοποιείται για την αλγοριθμική επεξεργασία του terrain. Συγκεκριμένα ο terrain terraform editor παρέχει μια σειρά τελεστών και φίλτρων που αν εφαρμοστούν στο αρχικό terrain τροποποιούν την μορφή του. Για την επιλογή textures για το terrain υπάρχουν τα εργαλεία **terrain texture editor** και **terrain texture painter**. Συγκεκριμένα μέσω του terrain texture editor καθορίζεται ο τρόπος με τον οποίο θα εφαρμοστεί το texture στο terrain, προσδίδοντας στην εμφάνιση του περισσότερες λεπτομέρειες. Ο terrain texture painter επιτρέπει την επιλογή μέχρι και 6 διαφορετικών textures τα οποία μπορούν να εφαρμοστούν σε διαφορετικά σημεία του terrain

Ο **GUI editor** είναι η εφαρμογή που επιτρέπει τον σχεδιασμό του γραφικού περιβάλλοντος το οποίο χρησιμοποιείται για την επικοινωνία του χρήστη με την εφαρμογή. Μέσω του GUI editor δίνεται η δυνατότητα κατασκευής αντικειμένων που αποτελούν συστατικά στοιχεία του γραφικού περιβάλλοντος. Επίσης είναι δυνατός ο προσδιορισμός της θέσης τους, του μεγέθους τους και των υπολοίπων χαρακτηριστικών τους όπως το αν θα είναι ορατά ή όχι καθώς και οι ενέργειες που θα εκτελούνται κατά την επιλογή τους, αν πρόκειται για αλληλεπιδραστικά αντικείμενα. Επίσης καθορίζεται το μέγεθος του παραθύρου της εφαρμογής και ο τρόπος με τον οποίο θα προσαρμοστεί το μέγεθος και η θέση των στοιχείων του GUI όταν μεταβάλλεται το μέγεθος του παραθύρου που τα περιέχει. Ο GUI editor παρέχει μια δενδρική δομή στην οποία εμφανίζονται όλα τα στοιχεία του γραφικού περιβάλλοντος. Με επιλογή κάποιου αντικειμένου της δενδρικής δομής εμφανίζονται οι ιδιότητες του αντικειμένου αυτού.

## 2.2: 3D Studio Max

Το 3D Studio Max είναι ένα πρόγραμμα δημιουργίας τρισδιάστατων μοντέλων, και χρησιμοποιήθηκε για την κατασκευή των κτιριακών δομών και αντικειμένων της εφαρμογής.

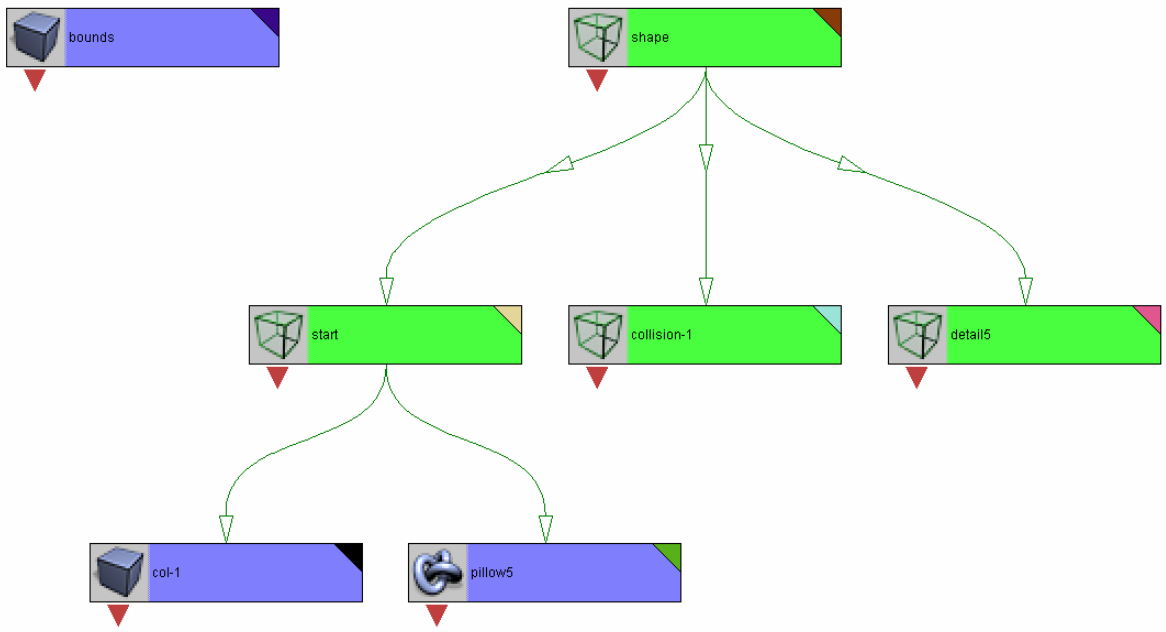
Τα αντικείμενα που σχεδιάζονται στο 3D Studio Max, όπως όλα τα τρισδιάστατα μοντέλα, αποτελούνται από ένα σύνολο σημείων-κορυφών και ένα σύνολο τριγωνικών επιφανειών. Ειδικά στην περίπτωση των χαρακτήρων ή κινούμενων αντικειμένων υπάρχει και το σύνολο των κόμβων του σκελετού του αντικειμένου. Οι κορυφές και οι επιφάνειες που σχηματίζουν ένα αντικείμενο μπορούν να ομαδοποιηθούν για την διευκόλυνση του σχεδιαστή.

Ο σχεδιασμός των αντικειμένων πραγματοποιείται με χρήση των εργαλείων που παρέχει το πρόγραμμα και αφορούν στην κατασκευή και επιλογή κορυφών, επιφανειών, κόμβων ή ομάδων, την κίνηση τους, την περιστροφή τους, τον προσδιορισμό του μεγέθους τους, την διαίρεση ή την συνένωσή τους. Επίσης παρέχεται η δυνατότητα άμεσης κατασκευής απλών σχημάτων όπως ορθογωνίων παραλληλεπιπέδων, σφαιρών, κυλίνδρων ή απλών επιφανειών. Μερικές βασικές λειτουργίες που περιλαμβάνει το πρόγραμμα για την διαχείριση των κορυφών είναι η συνένωση δύο ή περισσότερων κορυφών, η δημιουργία κορυφής πάνω σε μία ακμή με σκοπό την διαίρεση της ακμής, καθώς και η μετακίνηση και προσκόλληση μίας κορυφής πάνω σε συγκεκριμένη επιφάνεια. Αντίστοιχα για τον χειρισμό των επιφανειών παρέχονται επιλογές όπως η διαίρεση της επιφάνειας, η εναλλαγή εσωτερικής και εξωτερικής όψης και η αντιστροφή της σειράς των κορυφών που την απαρτίζουν. Επίσης είναι δυνατή η προσωρινή απόκρυψη μίας επιφάνειας ή ακόμα και ολόκληρης ομάδας για την διευκόλυνση της μοντελοποίησης.

Για κάθε texture που αντιστοιχίζεται σε κάποια επιφάνεια του μοντέλου δημιουργείται ένα νέο material στο 3D Studio Max. Τα textures είναι εικόνες JPG ή PNG που καλύπτουν τις επιφάνειες του μοντέλου, δίνοντας του μια «υφή». Οι εικόνες που χρησιμοποιούνται σαν textures δημιουργούνται με το PhotoImpact και πρέπει να έχουν διαστάσεις πολλαπλάσια του 2 (π.χ. 64x64, 128x512 κτλ).

Το 3D Studio Max επιτρέπει την ταυτόχρονη επισκόπηση του σχεδιαζόμενου αντικειμένου από διαφορετικές όψεις, μέσω πολλαπλών παραθύρων. Παραδοσιακά το 3D Studio Max έχει 4 παράθυρα με όψεις top, front, left, perspective.

Καθώς τα μοντέλα του 3D Studio Max σώζονται σαν .max αρχεία ενώ τα αρχεία που χρησιμοποιεί το Torque Game Engine είναι .dts είναι προφανές ότι απαιτείται ένα πρόγραμμα-plugin για το 3DSMax που θα επιτρέπει την αλλαγή του αρχείου σε .dts. Το πρόγραμμα αυτό είναι το Max2DTS και απαιτεί από το χρήστη μερικές ρυθμίσεις για να λειτουργήσει. Κατ' αρχήν απαιτεί μια ιεραρχία από dummy objects: το shape είναι η ρίζα με παιδιά το detailX όπου X το επίπεδο λεπτομέρειας που μας ενδιαφέρει (το μοντέλο μπορεί να έχει πολλά επίπεδα λεπτομέρειας, έτσι ώστε όταν είναι μακριά να αποκτά ένα απλούστερο σχήμα για να μειώνει τις απαιτήσεις από τον υπολογιστή), το collision-1 το οποίο καταδεικνύει ότι υπάρχει collision mesh με αριθμό -1, και τέλος το shape στο οποίο προσαρτώνται τα μοντέλα (το col-1 που είναι το collision mesh και το nameX όπου name είναι το όνομα του αντικειμένου και X το επίπεδο λεπτομέρειας). Τέλος υπάρχει ένα ορθογώνιο μοντέλο bounds το οποίο περικλείει τα δύο μοντέλα (αντικείμενο και collision mesh) και το οποίο δεν προσαρτάται σε κανένα dummy object. Αυτό το αντικείμενο είναι το «περίβλημα» του μοντέλου, και είναι απαραίτητο για όλα τα DTS μοντέλα.




**ΕΙΚΟΝΑ 1:** Η ΙΕΡΑΡΧΙΑ ΑΝΤΙΚΕΙΜΕΝΩΝ ΚΑΙ DUMMY OBJECTS ΕΝΟΣ DTS ΑΝΤΙΚΕΙΜΕΝΟΥ

Το collision mesh είναι στην ουσία η σκληρή επιφάνεια του μοντέλου, πάνω στην οποία μπορεί να βαδίσει ο χρήστης ή μέσω της οποίας εμποδίζεται ο χρήστης να κινηθεί έξω από τα όρια του κτηρίου. Επίσης μέσω του collision mesh μπορεί η μηχανή να εξετάζει ποιο αντικείμενο εξετάζει ή ενεργοποιεί ο χρήστης. Ένα αντικείμενο χωρίς collision mesh επιτρέπει στον χρήστη να περάσει από μέσα του, ενώ παράλληλα ο χρήστης δεν μπορεί να εξετάσει ή να ενεργοποιήσει το εν λόγω αντικείμενο. Πολλά από τα μικρότερα αντικείμενα έχουν απλά ένα ορθογώνιο για collision mesh (περίπου στο μέγεθος του αντικειμένου) αλλά τα κτήρια πρέπει να έχουν ένα προσεγμένο collision mesh αλλά παράλληλα και απλό έτσι ώστε να μειώνονται οι απαιτήσεις υπολογιστικής ισχύος.

Εκτός από την μοντελοποίηση αντικειμένων το 3D Studio Max παρέχει και την δυνατότητα σχεδιασμού της κίνησης των χαρακτήρων και των κινούμενων αντικειμένων. Συγκεκριμένα με χρήση του animation bar στο κάτω μέρος του παραθύρου της εφαρμογής διατίθεται στον χρήστη μία σειρά frames που μπορούν να χρησιμοποιηθούν για το σχεδιασμό της κίνησης. Έτσι μπορούν να οριστούν keyframes μέσω των οποίων περιγράφεται η κίνηση. Δύο keyframes για την αρχή και το τέλος της κίνησης είναι αρκετά για τα περισσότερα animations των αντικειμένων της εφαρμογής. Η σχετική με την κίνηση πληροφορία μπορεί είτε να ενσωματωθεί στο αρχείο .dts που θα προκύψει, ή να αποθηκευτεί σε ξεχωριστά αρχεία dsq. Αν επιλεγεί η δημιουργία αρχείων dsq κατασκευάζεται ένα αρχείο για κάθε ακολουθία κίνησης, τα οποία κατόπιν συντίθεται σε ένα dts μέσα από τον κώδικα TorqueScript (ένα παράδειγμα τέτοιας σύνθεσης βρίσκεται στο **Κεφάλαιο 7: Τεχνητή Νοημοσύνη**). Για την ικανοποιητική μεταφορά του αντικειμένου μαζί με το animation του, πρέπει να τοποθετηθεί ένα ειδικό αντικείμενο (που βρίσκεται στον Max2DTS exporter) το οποίο θα περιγράφει το όνομα του sequence και τα keyframes αρχής και τέλους της. Μπορούν να τοποθετηθούν πολλά sequences στο ίδιο αντικείμενο, αρκεί να έχουν διαφορετικό όνομα.

## 2.3: Ulead PhotoImpact

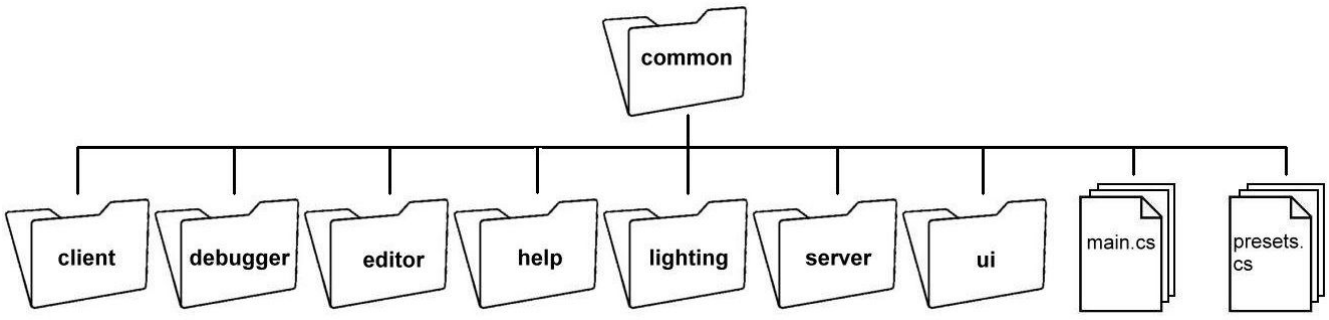
Το PhotoImpact είναι ένα πρόγραμμα επεξεργασίας εικόνας, και χρησιμοποιήθηκε για την επεξεργασία των textures των κτιριακών δομών και αντικειμένων που εμφανίζονται στον εικονικό κόσμο. Περιλαμβάνει ένα πλήθος εργαλείων που χρησιμοποιούνται για την κατασκευή και επεξεργασία εικόνων. Τα εργαλεία αυτά επιτρέπουν την επιλογή και απομόνωση συγκεκριμένων τμημάτων της εικόνας, την σχεδίαση γραμμών, τον χρωματισμό περιοχών και την επιλογή χρωματικών αποχρώσεων από έναν κατάλογο χρωμάτων, από το χρωματικό φάσμα ή από συγκεκριμένες περιοχές της εικόνας. Επίσης το πρόγραμμα παρέχει εργαλεία που επιτρέπουν την εισαγωγή γεωμετρικών σχημάτων, την αντιγραφή περιοχών μίας εικόνας, την εισαγωγή κειμένου, την επιλογή γραμματοσειράς και μία σειρά άλλων επιλογών που συμβάλλουν στην διαμόρφωση της εικόνας.
Με το PhotoImpact υπάρχει η δυνατότητα δημιουργίας ξεχωριστών περιοχων/αντικειμένων μέσα σε αυτή. Κάθε τέτοιο αντικείμενο μπορεί μετακινηθεί ή να τροποποιηθεί ξεχωριστά και ανεξάρτητα από την υπόλοιπη εικόνα.
Επιπλέον το πρόγραμμα περιέχει πληθώρα έτοιμων φίλτρων που μπορούν να εφαρμοστούν σε μία εικόνα ή στα αντικείμενα αυτής. Η μορφή της εικόνας τροποποιείται με προσθήκη θορύβου ή με αλλαγή χαρακτηριστικών της στοιχείων (brightness, saturation κτλ).
Το είδος της εικόνας και η ανάλυση της καθορίζεται από τον χρήστη. Έτσι όσον αφορά στο είδος της εικόνας ο χρήστης μπορεί να επιλέξει αν επιθυμεί να κατασκευάσει μια εικόνα Bitmap, Grayscale, RGB Color, CMYC Color ή Lab Color. Περιορισμοί στο περιβάλλον Torque επιτρέπουν μόνο μερικά formats (πιο συχνά χρησιμοποιούνται JPG και PNG) και όλες οι εικόνες πρέπει να έχουν διαστάσεις πολλαπλάσια του 2 (π.χ. 64x64, 512x128 κτλ).
Τέλος το πρόγραμμα παρέχει εργαλεία για την μετακίνηση ή περιστροφή της εικόνας ή τμημάτων της, και την ρύθμιση χαρακτηριστικών όπως η φωτεινότητα και το χρώμα. Επίσης παρέχονται πολλές επιπλέον επιλογές και δυνατότητες που χρησιμοποιούνται για την διαμόρφωση της τελικής μορφής μίας εικόνας σύμφωνα με τις επιθυμίες του χρήστη.

# ΚΕΦΑΛΑΙΟ 3:
# Δομή μιας εφαρμογής Torque

Για μια εφαρμογή γραμμένη σε Torque μπορεί να χρησιμοποιηθεί ικανοποιητικά οποιαδήποτε δομή φακέλων. Η εφαρμογή θα περιλαμβάνει τα αρχεία .mis που περιγράφουν τον ψηφιακό κόσμο, τις εικόνες και τα στοιχεία του GUI, τα τρισδιάστατα μοντέλα, τα αρχεία ήχου, και αρχεία κώδικα που περιγράφουν πως συμπεριφέρονται τα αντικείμενα και οι NPCs της εφαρμογής. Ο μόνος πραγματικός περιορισμός στο πώς δομούνται οι φάκελοι σε μια εφαρμογή Torque είναι ότι το αρχείο main.cs πρέπει να βρίσκεται στον ίδιο φάκελο με το εκτελέσιμο πρόγραμμα (TGE.exe), και αυτός ο φάκελος θα είναι ο φάκελος ρίζας (root folder) της εφαρμογής. Το ελάχιστο δυνατό για να είναι λογικά οργανωμένη η δομή της εφαρμογής είναι να υπάρχει ένας φάκελος που περιέχει τον κοινό κώδικα, τον κώδικα που θα ήταν ουσιαστικά ο ίδιος μεταξύ των διάφορων εφαρμογών και των παραλλαγών τους, και έναν άλλο φάκελο που θα περιέχει τον κώδικα ελέγχου και τους πόρους που αναφέρονται σε μια συγκεκριμένη εφαρμογή. Η GarageGames χρησιμοποιεί αυτά τα δύο βασικά υπόδεντρα, common και control, στις ενδεικτικές εφαρμογές της, και η δομή αυτή διατηρήθηκε και για την τρέχουσα εφαρμογή.

Τα αρχεία κώδικα για το TorqueScript έχουν την επέκταση cs. Αφότου τα αρχεία κώδικα γίνουν compiled, αποκτούν την επέκταση .cs.dso. Δεν υπάρχει κανένας τρόπος να επανέλθει ένα αρχείο cs.dso σε ένα αρχείο .cs. Όταν καλείται το TGE.exe, ψάχνει το αρχείο main.cs που βρίσκεται στον φάκελο ρίζας, στον οποίο πρέπει να υπάρχουν πληροφορίες για όλα τα αρχεία που θα χρησιμοποιηθούν στο πρόγραμμα και σε ποιον φάκελο βρίσκονται. Το main.cs είναι το μόνο αρχείο που καλείται αυτόματα, και αυτό το κάνει ιδιαίτερα σημαντικό. Στην παρούσα εφαρμογή το main.cs καλεί δύο αρχεία, το control/main.cs και το common/main.cs, δηλαδή τα αντίστοιχα αρχεία των δύο υποφακέλων (common και control) μέσα από τα οποία καλούνται με τη σειρά τους άλλα αρχεία που βρίσκονται στους υποφακέλους αυτούς.

## 3.1: Ο φάκελος common




**ΕΙΚΟΝΑ 2:** ΔΟΜΗ ΤΟΥ ΦΑΚΕΛΟΥ COMMON

Ο φάκελος common περιέχει όλες τις απαραίτητες μεταβλητές και συναρτήσεις που ορίζουν τον γενικό τρόπο παρουσίασης και λειτουργίας εφαρμογών Torque. Ο φάκελος common συνήθως δεν διαφέρει από εφαρμογή σε εφαρμογή, αφού τα χαρακτηριστικά της εκάστοτε εφαρμογής ορίζονται στον φάκελο control. Αυτό δε σημαίνει βέβαια ότι δεν μπορεί κάποιος να αλλάξει τον κώδικα των συναρτήσεων του φακέλου common, προκειμένου να αποκτήσει πρόσβαση σε πολλές από τις λειτουργικές δομές του Torque.

Θα ρίξουμε μια σύντομη ματιά στις κύριες συναρτήσεις του φακέλου common, παρόλο που για την εφαρμογή μας δεν έχουμε τροποποιήσει καμία από αυτές.

Δύο βασικά πράγματα που συμβαίνουν κατά την έναρξη μιας εφαρμογής Torque είναι οι κλήσεις των ***InitBaseServer*** και ***InitBaseClient***, και οι δύο από τις οποίες ορίζονται στο common/main.cs. Αυτές είναι καίριες λειτουργίες, αν και ουσιαστικά απλά καλούν άλλα αρχεία .cs στους αντίστοιχους φακέλους (common/client για την InitBaseClient και common/server για την InitBaseServer).

### *3.1.1: Ο φάκελος common/server*

**Server.cs**

Το αρχείο **server.cs** είναι σημαντικό module για τη λειτουργία του εξυπηρετητή μιας εφαρμογής Torque. Περιέχει τις συναρτήσεις:

- *PortInit*
- *CreateServer*
- *DestroyServer*
- *ResetServerDefaults*
- *AddToServerGuidList*
- *RemoveFromServerGuidList*
- *OnServerInfoQuery*

Η ***PortInit*** προσπαθεί να πάρει τον έλεγχο μιας θύρας TCP/IP, και εάν δεν μπορεί, αυξάνει τον αριθμό θύρας έως ότου βρίσκει μια ανοιχτή θύρα που να μπορεί να χρησιμοποιήσει.
Η ***CreateServer*** προφανώς δημιουργεί έναν εξυπηρετητή για τον πελάτη. Ξεκινά όμως με μια κλήση στη συνάρτηση DestroyServer, η οποία απελευθερώνει και θέτει εκτός λειτουργίας τους πόρους, αν αυτοί υπάρχουν. Πρέπει να διευκρινιστεί ο τύπος εξυπηρετητή (single ή multi-user) και το όνομα της αποστολής (mission) που θα χρησιμοποιηθεί (περισσότερα για τις αποστολές στο **Κεφάλαιο 5: Κατασκευή του Εικονικού Κόσμου**). Η λειτουργία PortInit καλείται από εδώ, εάν ο εξυπηρετητής θα είναι για multi-user εφαρμογή. Το τελευταίο πράγμα που κάνει η CreateServer είναι η κλήση της συνάρτησης LoadMission, η οποία ξεκινά μια μακρά αλυσίδα γεγονότων που θα καλύψουμε παρακάτω.
Η ***DestroyServer*** απελευθερώνει και θέτει εκτός λειτουργίας πόρους και μηχανισμούς της εφαρμογής. Επίσης αποτρέπει να συμβούν περαιτέρω συνδέσεις και κλείνει τις ήδη υπάρχουσες, διαγράφει όλα τα αντικείμενα του εξυπηρετητή στις δομές MissionGroup, MissionCleanup, και ServerGroup και εκκαθαρίζει όλα τα DataBlocks από τη μνήμη.
Η ***ResetServerDefaults*** προφανώς επαναφορτώνει τα αρχεία στα οποία έχουν αποθηκευτεί οι αρχικές τιμές του εξυπηρετητή.
Οι ***AddToServerGuidList*** και ***RemoveFromServerGuidList*** είναι δύο συναρτήσεις για τη διαχείριση του καταλόγου των πελατών που συνδέονται με τον εξυπηρετητή.
Η ***OnServerInfoQuery*** διαχειρίζεται μηνύματα «ερώτησης» (query) από έναν κεντρικό εξυπηρετητή, επιστρέφοντας μόνο ένα μήνυμα ομαλής λειτουργίας. Ο κεντρικός εξυπηρετητής, εάν υπάρχει, θα δει αυτό το μήνυμα και θα συμπεράνει ότι ο εξυπηρετητής λειτουργεί. Αν ο εξυπηρετητής δεν λειτουργεί σωστά, δε θα επιστρέψει τέτοιο μήνυμα, οπότε ο κεντρικός εξυπηρετητής κάποια στιγμή θα σταματήσει να περιμένει για απάντηση και θα λάβει τα απαραίτητα μέτρα.

**Message.cs**

Όπως είναι προφανές από τον τίτλο του, αυτό το αρχείο ασχολείται με την επικοινωνία (in-game chat) των πελατών. Περιέχει τις συναρτήσεις:

- *MessageClient*
- *MessageTeam*
- *MessageTeamExcept*
- *MessageAll*
- *MessageAllExcept*
- *ChatMessageClient*
- *ChatMessageTeam*
- *ChatMessageAll*
- *SpamAlert*
- *GameConnection::SpamMessageTimeout*
- *GameConnection::SpamReset*

Οι πρώτες πέντε λειτουργίες είναι για την αποστολή των μηνυμάτων από τον εξυπηρετητή σε κάθε πελάτη ξεχωριστά, σε όλους τους πελάτες σε μια ομάδα, και σε όλους τους πελάτες σε μια εφαρμογή. Σ' αυτά προστίθενται και τα μηνύματα «εξαίρεσης» τα οποία στέλνονται σε όλους τους πελάτες (σε ομάδα ή σε εφαρμογή) εκτός από κάποιους συγκεκριμένους.

Ακολουθούν οι τρεις συναρτήσεις μηνυμάτων chat. Αυτές συνδέονται στο γραφικό περιβάλλον του chat που οι πελάτες θα χρησιμοποιούν για να επικοινωνήσουν μεταξύ τους. Όλες αυτές οι συναρτήσεις χρησιμοποιούν το τη συνάρτηση CommandToServer, άρα θα πρέπει να υπάρχουν χειριστές μηνυμάτων για αυτές τις εντολές και από την πλευρά του πελάτη.

Οι τρεις συναρτήσεις ελέγχου spam χρησιμοποιούνται από κοινού με τις συναρτήσεις μηνυμάτων chat. Η ***SpamAlert*** καλείται αμέσως πριν υποβληθεί σε επεξεργασία κάθε εξερχόμενο μήνυμα συνομιλίας για την αποστολή. Σκοπός της είναι να ανιχνεύσει εάν ένας φορέας γεμίζει το παράθυρο συνομιλίας με τα μηνύματα (γνωστό και ως spamming). Εάν υπάρχουν πάρα πολλά μηνύματα σε ένα σύντομο χρονικό διάστημα (που καθορίζεται από τη συνάρτηση ***SpamMessageTimeout***), παύουν να στέλνονται άλλα μηνύματα, και στέλνεται στον πελάτη ένα προειδοποιητικό μήνυμα. Η ***SpamReset*** επαναφέρει τον πελάτης-spammer σε κανονική λειτουργία αποστολής μηνυμάτων μετά από κάποιο χρόνο.

**MissionLoad.cs**

Το Torque ορίζει μια αποστολή (mission) σαν μια παραδοσιακή πίστα παιχνιδιού. Μια αποστολή ορίζεται και φυλάσσεται σε ένα αρχείο .mis που περιέχει ορισμούς για τα διάφορα αντικείμενα αλλά και τις συντεταγμένες τους μέσα στην αποστολή. Οι αποστολές μεταφορτώνονται από τον εξυπηρετητή στον πελάτη τη στιγμή που ξεκινά η αποστολή ή όταν ένας πελάτης συνδέεται σε μια αποστολή σε εξέλιξη. Κατ' αυτό τον τρόπο ο εξυπηρετητής έχει τον απόλυτο έλεγχο όσων βλέπει και χρησιμοποιεί ο πελάτης στην αποστολή.
Το αρχείο MissionLoad.cs περιέχει τις συναρτήσεις:

- LoadMission
- LoadMissionStage2
- EndMission
- ResetMission

Η συνάρτηση ***LoadMission*** καλείται όπως είδαμε από τη συνάρτηση createServer, και ξεκινά τη διαδικασία φόρτωσης μιας αποστολής από τον εξυπηρετητή. Οι πληροφορίες της αποστολής συγκεντρώνονται από το αρχείο .mis και στέλνονται σε όλους τους πελάτες.

Κατόπιν καλείται η συνάρτηση ***LoadMissionStage2***, με την οποία ο υπολογιστής υπολογίζει την CRC τιμή (Cyclic Redundancy Check) της αποστολής. Έτσι εξασφαλίζεται ότι δεν υπήρχε απώλεια πληροφορίας κατά την διανομή της αποστολής στους πελάτες, και ότι δεν έχει «πειράξει» τα αρχεία της αποστολής κάποιος τρίτος.

Μόλις φορτωθεί η αποστολή στον εξυπηρετητή, στέλνεται σε κάθε πελάτη η αποστολή μέσω μιας κλήσης της συνάρτησης LoadMission του αντικειμένου GameConnection.

Η ***EndMission*** απελευθερώνει τους πόρους και θέτει εκτός λειτουργίας μηχανισμούς σχετικούς με την τρέχουσα αποστολή, αδειάζοντας τον εξυπηρετητή για να φορτώσει μια νέα αποστολή όταν του ζητηθεί.

Η ***ResetMission*** προετοιμάζει τον εξυπηρετητή για μια νέα αποστολή αν χρησιμοποιείται η τεχνική «ανακύκλωσης» των αποστολών (mission-cycling).

**MissionDownload.cs**

Το αρχείο MissionDownload.cs διαχειρίζεται τις διαδικασίες μεταφόρτωσης των αποστολών από την πλευρά του εξυπηρετητή, και περιέχει τις συναρτήσεις:

- GameConnection::LoadMission
- GameConnection::OnDataBlocksDone
- GameConnection::ClientWantsGhostAlwaysRetry
- GameConnection::OnGhostAlwaysFailed
- GameConnection::OnGhostAlwaysObjectsReceived
- ServerCmdMissionStartPhase1Ack
- ServerCmdMissionStartPhase2Ack
- ServerCmdMissionStartPhase3Ack

Το αρχείο περιέχει τις συναρτήσεις για τη μεταφόρτωση του αντικειμένου GameConnection για κάθε πελάτη. Η εκκίνηση της μεταφόρτωσης ξεκινάει όταν καλείται η συνάρτηση LoadMission στο τέλος της συνάρτησης LoadMissionStage2 στο αρχείο MissionLoad.cs του εξυπηρετητή. Ακολουθεί μια συγχρονισμένη σειρά ενεργειών μεταξύ εξυπηρετητή και πελάτη με κλήσεις των εντολών CommandToServer και CommandToClient. Ο εξυπηρετητής ζητά από τον πελάτη την άδεια να ξεκινήσει να μεταφορτώνει την αποστολή με την εντολή ***StartPhaseN*** (με N=1,2,3 ανάλογα με τη φάση της μεταφόρτωσης). Όταν ο πελάτης είναι έτοιμος, στέλνει το μήνυμα ***StartPhaseNAck*** με χρήση της συνάρτησης CommandToServer.

Η ***onDataBlocksDone*** καλείται όταν τελειώνει η 1η φάση, δηλαδή όταν ολοκληρώνεται η μεταφόρτωση των Datablocks. Η συνάρτηση αυτή ξεκινά τη 2η φάση στέλνοντας την εντολή MissionStartPhase2 στον πελάτη.

Η ***onGhostAlwaysObjects*** καλείται όταν τελειώνει η 2η φάση. Καθώς τελειώνει η 2η φάση, ο πελάτης έχει όλες τις απαραίτητες πληροφορίες για να προσομοιώσει τα δυναμικά αντικείμενα του εξυπηρετητή που είναι αντιγραμμένα στους πελάτες. Κατόπιν η συνάρτηση αυτή στέλνει την εντολή MissionStartPhase3 στον πελάτη, για να εκκινήσει η 3η φάση.

Τέλος, όταν ο εξυπηρετητής λαμβάνει το μήνυμα StartPhase3Ack, ξεκινά την αποστολή εισάγοντας τον πελάτη σε αυτήν.

**ClientControl.cs**

Το αρχείο ClientControl.cs διαχειρίζεται τις περισσότερες διαδικασίες (από την πλευρά του εξυπηρετητή) με τις οποίες ο εξυπηρετητής επικοινωνεί με τον πελάτη, και περιέχει τις συναρτήσεις:

- GameConnection::OnConnectRequest
- GameConnection::OnConnect
- GameConnection::SetPlayerName
- IsNameUnique
- GameConnection::OnDrop
- GameConnection::StartMission
- GameConnection::EndMission
- GameConnection::SyncClock
- GameConnection::IncScore

Η συνάρτηση ***onConnectRequest*** είναι ο αποδέκτης από την πλευρά του εξυπηρετητή του μηνύματος GameConnection::Connect. Η συνάρτηση αυτή εξετάζει την αίτηση του πελάτη να συνδεθεί (ελέγχει αν ο πελάτης είναι σε BanList, αν ο εξυπηρετητής είναι γεμάτος κ.α.). Αν το αίτημα του πελάτη να συνδεθεί γίνεται αποδεκτό από τον εξυπηρετητή, η onConnectRequest επιστρέφει μια κενή σειρά χαρακτήρων.

Η συνάρτηση ***onConnect*** καλείται αφού εγκρίνει ο εξυπηρετητής το αίτημα σύνδεσης. Παράμετροι που δέχεται η onConnect είναι ένα handle του πελάτη και το όνομά του. Η συνάρτηση ξεκινά στέλνοντας πληροφορίες της αποστολής στον πελάτη για να εμφανίζει στον χρήστη ενώ φορτώνεται η αποστολή. Αν ο πελάτης είναι και host, η συνάρτηση θέτει τον πελάτη σαν superAdmin. Κατόπιν προστίθεται ο πελάτης σε μια λίστα ID χρηστών που διατηρεί ο εξυπηρετητής, αρχικοποιούνται διάφορες ρυθμίσεις για την εφαρμογή, και στέλνεται μήνυμα εισαγωγής του νέου πελάτη στους ήδη συνδεδεμένους χρήστες. Τέλος ξεκινά να μεταφορτώνεται η αποστολή στον πελάτη, όπως αναλύσαμε παραπάνω.

Η συνάρτηση ***setPlayerName*** ελέγχει και τροποποιεί το όνομα του πελάτη έτσι όπως εστάλη στην onConnect. Οι τροποποιήσεις περιλαμβάνουν έλεγχο μήκους χαρακτήρων, κενών ή μη επιτρεπτών χαρακτήρων κ.α. Τέλος με χρήση της ***isNameUnique*** ελέγχεται αν το όνομα υπάρχει ήδη στη λίστα των πελατών που διατηρεί ο εξυπηρετητής.

Η ***onDrop*** καλείται όταν γίνεται η επιλογή να αποβληθεί ένας πελάτη. Η συνάρτηση αυτή στέλνει μηνύματα στον πελάτη που αποβάλλεται αλλά και σε όλους τους άλλους συνδεδεμένους χρήστες. Αν αποβληθεί και ο τελευταίος συνδεδεμένος πελάτης, η συνάρτηση αυτή επανεκκινεί τον εξυπηρετητή.

Η συνάρτηση ***StartMission*** απλά ενημερώνει τους συνδεδεμένους πελάτες ότι ξεκινά μια αποστολή, ενώ αντίστοιχα η ***EndMission*** ότι η αποστολή τελείωσε. Η συνάρτηση SyncClock ελέγχει αν τα «ρολόγια» όλων των πελατών είναι συγχρονισμένα, και πρέπει να κληθεί αφού φορτωθεί η αποστολή αλλά πριν τοποθετηθεί το avatar του πελάτη.

Τέλος, η μέθοδος incScore καλείται όταν ο χρήστης πρέπει να επιβραβευθεί με αύξηση του score του. Όταν ένας χρήστης αυξήσει το score του, όλοι οι άλλοι πελάτες ενημερώνονται.

**Game.cs**

Το αρχείο Game.cs από την πλευρά του εξυπηρετητή περιέχει τις περισσότερες πληροφορίες για τον τρόπο που τρέχει μια εφαρμογή. Περιέχει τις εξής συναρτήσεις:

- OnServerCreated
- OnServerDestroyed
- OnMissionLoaded
- OnMissionEnded
- OnMissionReset
- StartGame
- EndGame
- GameConnection::OnClientEnterGame
- GameConnection::OnClientLeaveGame

Η συνάρτηση ***onServerCreated*** καλείται μόλις δημιουργηθεί ο εξυπηρετητής, και είναι το κατάλληλο σημείο για να φορτωθούν dataBlocks της πλευράς του εξυπηρετητή. Όσα υλοποιεί η onServerCreated πρέπει να αναιρεθούν στην ***onServerDestroyed***, που καλείται πριν τερματιστεί ο εξυπηρετητής.
Η συνάρτηση ***onMissionLoaded*** καλείται όταν η αποστολή έχει φορτωθεί, και εδώ μπορούν να τοποθετηθούν ειδικές συνθήκες αναλόγως την αποστολή κλπ. Ομοίως η ***onMissionEnded*** καλείται πριν τερματιστεί η αποστολή και αναιρεί όλα όσα υλοποίησε η onMissionLoaded. Τέλος η ***onMissionReset*** καλείται όταν δίνεται εντολή επανεκκίνησης της αποστολής.
Οι συναρτήσεις ***onClientEnterGame*** και ***onClientLeaveGame*** καλούνται όταν ένας πελάτης ξεκινά μια αποστολή και όταν φεύγει από μια αντίστοιχα.

### *3.1.2: Ο φάκελος common/client*

**Canvas.cs**

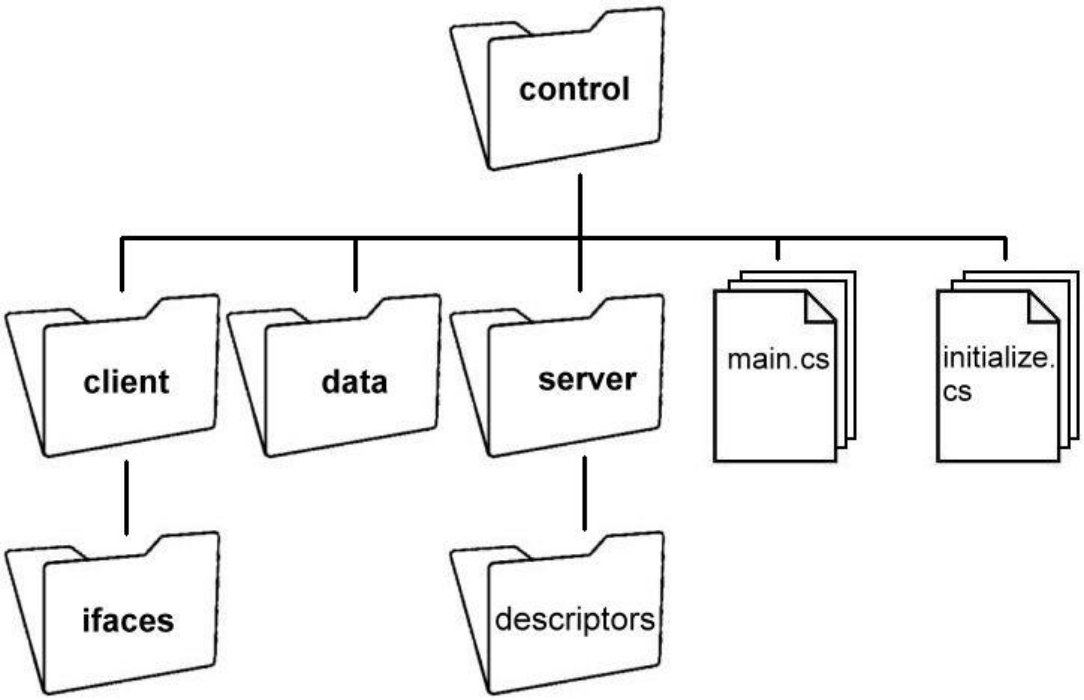
Το αρχείο Canvas.cs περιέχει πολλές σημαντικές συναρτήσεις που ορίζουν το γραφικό περιβάλλον της εφαρμογής. Περιέχει την συνάρτηση ***initCanvas***, που φορτώνει διάφορα αρχεία γραφικού περιβάλλοντος του φακέλου common/ui. Η initCanvas δημιουργεί το Canvas, που είναι μια αόριστη αίτηση δημιουργίας ενός παραθύρου προς το Windows API, και αρχικοποιεί διάφορες ρυθμίσεις στο OpenGL και παγκόσμιες μεταβλητές. Επίσης εκτελεί διάφορα αρχεία στο common/ui που ορίζουν στοιχεία του γραφικού περιβάλλοντος.
Η εντολή ResetCanvas ελέγχει αν υπάρχει το αντικείμενο Canvas και αν ναι, απαιτεί να ξαναγίνει rendered.

**MissionDownload.cs**

Το MissionDownload.cs είναι συμπληρωματικό του ομώνυμου αρχείου στο φάκελο common/server, καθώς ορίζει πώς θα απαντά ο πελάτης στις εντολές μεταφόρτωσης αποστολής από τον εξυπηρετητή.
Στην 1η φάση, η συνάρτηση ClientCmdMissionStartPhase1 καλεί τη συνάρτηση onMissionDownloadPhase1 που ορίζεται στον φάκελο control, παρακάτω. Ο βασικός σκοπός της είναι να εκτυπώνεται κάτι στην οθόνη ενώ φορτώνονται τα διάφορα DataBlocks. Όταν ολοκληρωθεί η κλήση αυτή, στέλνεται μήνυμα MissionStartPhase1Ack στον εξυπηρετητή, με χρήση της CommandToServer. Όμοια πράττουν και οι ClientCmdMissionStartPhase2 και ClientCmdMissionStartPhase3, με αντίστοιχες κλήσεις συναρτήσεων.

Η συνάρτηση onDataBlockObjectReceived καλείται κάθε φορά που ο πελάτης δέχεται ένα DataBlock, και καλεί με τη σειρά της την συνάρτηση onPhase1Progress που ορίζεται στον φάκελο control, παρακάτω.
Η συνάρτηση ***onGhostAlwaysStarted*** ελέγχει τον αριθμό αντικειμένων που είναι ghosted, μετά το τέλος της μεταφόρτωσης της 2ης φάσης.
Η συνάρτηση ***LightScene*** προσθέτει φώτα και κάνει render τον ψηφιακό κόσμο στη μηχανή γραφικών, επιστρέφοντας στον εξυπηρετητή το μήνυμα ***SceneLightingComplete*** όταν τελειώσει.

## 3.2: Ο φάκελος control




**ΕΙΚΟΝΑ 3:** ΔΟΜΗ ΤΟΥ ΦΑΚΕΛΟΥ CONTROL

### *3.2.1: Ο φάκελος control/data*

Στον φάκελο data αποθηκεύονται όλα τα μοντέλα της εφαρμογής, μαζί με τα αντίστοιχα textures. Επίσης ο φάκελος data περιέχει το αρχείο αποστολής museum_map.mis στο οποίο καταχωρούνται τα περισσότερα αντικείμενα της αποστολής. Τα περιεχόμενα του φακέλου data είναι ως επί το πλείστον μοντέλα και ως εκ τούτου η περιγραφή τους ξεφεύγει από το σκοπό της αναφοράς. Παρ' όλ' αυτά η αποστολή museum_map.mis αναλύεται στο **Κεφάλαιο 5: Κατασκευή του ψηφιακού κόσμου** ενώ το μοντέλο και τα animations του Bot-ξεναγού περιγράφονται στο **Κεφάλαιο 7: Τεχνητή νοημοσύνη**.

### *3.2.2: Ο φάκελος control/client*

Ο φάκελος αυτός περιέχει αρχεία που αφορούν το γραφικό περιβάλλον της εφαρμογής, δηλαδή μεταξύ άλλων τα εικονίδια του cursor, τις εικόνες που χρησιμοποιούνται ως SplashScreens, τα fonts που χρησιμοποιεί η εφαρμογή, την περιγραφή του playerInterface, καθώς και οι εντολές που στέλνει ο εξυπηρετητής στον πελάτη. Όλα τα παραπάνω θα περιγραφούν εκτενώς στο **Κεφάλαιο 4: User Interaction**.

### *3.2.3: Ο φάκελος control/server*

Ο φάκελος αυτός περιέχει αρχεία που ορίζουν τα μηνύματα που στέλνει ο πελάτης στον εξυπηρετητή και ποιες συναρτήσεις πυροδοτούν. Αυτά τα αρχεία περιγράφονται στο **Κεφάλαιο 2: User Interaction**. Ο φάκελος control/server περιέχει και τον φάκελο descriptors όπου βρίσκονται τα αρχεία που περιγράφουν τα αλληλεπιδραστικά αντικείμενα και τους τρόπους που αντιδρούν στις ενέργειες του χρήστη. Μεταξύ αυτών υπάρχει και το αρχείο bot.cs στο οποίο περιγράφεται η συμπεριφορά του Bot-ξεναγού που θα περιγραφεί στο **Κεφάλαιο 7: Τεχνητή Νοημοσύνη**. Τα υπόλοιπα αλληλεπιδραστικά αντικείμενα που περιέχονται στο φάκελο descriptors θα περιγραφούν στο **Κεφάλαιο 6: Αλληλεπιδραστικά Αντικείμενα**.

# ΚΕΦΑΛΑΙΟ 4:
# Αλληλεπίδραση με τον Χρήστη

## 4.1: Εισαγωγικές οθόνες

Πολλές εφαρμογές χρησιμοποιούν splashscreens για να ειδοποιούν το χρήστη ότι το πρόγραμμα είναι στο στάδιο της φόρτωσης. Με άλλα λόγια παρέχουν ενδείξεις για μια χρονοβόρα διαδικασία εν εξελίξει.

Καθώς ο χρόνος φόρτωσης της εφαρμογής μας είναι ιδιαίτερα μεγάλος, απαιτείται να χρησιμοποιήσουμε κι εμείς κάποιες οθόνες κατά τη φάση αυτή. Επιλέξαμε δύο οθόνες για την εισαγωγή στην εφαρμογή. Η πρώτη οθόνη αναφέρει το δημιουργό, ενώ η επόμενη τον τίτλο της εφαρμογής. Η εναλλαγή μεταξύ των οθονών γίνεται με το πάτημα οποιουδήποτε πλήκτρου (πληκτρολογίου ή ποντικιού).

Η πρώτη οθόνη είναι ουσιαστικά απλά μια εικόνα εισαγωγική που αναφέρει το δημιουργό της εφαρμογής. Για την περιγραφή της οθόνης και του γραφικού περιβάλλοντος χρησιμοποιούμε το αρχείο ***Splash1.gui***, τα περιεχόμενα του οποίου είναι τα παρακάτω:

```
//--- OBJECT WRITE BEGIN ---
new GuiChunkedBitmapCtrl(SplashScreen1) {
   profile = "GuiSplashProfile";
   horizSizing = "width";
   vertSizing = "height";
   position = "0 0";
   extent = "640 480";
   minExtent = "8 8";
   visible = "1";
   helpTag = "0";
   bitmap = "./splash1";
   useVariable = "0";
   tile = "0";
   noCursor=1;
   new GuiInputCtrl(SplashScreenInputCtrl) {
      profile = "GuiInputCtrlProfile";
      position = "0 0";
      extent = "10 10";
   };
};
//--- OBJECT WRITE END ---
```

Όπως βλέπουμε, το μόνο στοιχείο του γραφικού περιβάλλοντος γι' αυτήν την οθόνη είναι μια εικόνα. Επιλέγεται το ***GuiChunkedBitmapCtrl*** γιατί η εικόνα μας είναι μεγάλη και με το format αυτό «σπάει» σε μικρότερα κομμάτια για να βελτιώσει την απόδοση στην οθόνη. Από τις μεταβλητές που περιέχονται στο αντικείμενο ***SplashScreen1*** οι σημαντικότερες είναι η position που ορίζει ότι η εικόνα ξεκινά από το πάνω αριστερά μέρος της οθόνης της εφαρμογής, η extent που ορίζει ότι η εικόνα θα εκτείνεται σε ολόκληρη την οθόνη, η bitmap που ορίζει πού βρίσκεται το αρχείο, η noCursor που πιστοποιεί ότι δε θα εμφανίζεται cursor στην οθόνη, και τέλος το αντικείμενο ***SplashScreenInputCtrl***, που ουσιαστικά είναι ένας χειριστής για οποιαδήποτε πάτημα πλήκτρου του χρήστη.

Η δεύτερη οθόνη περιέχει τον τίτλο της εφαρμογής και μια χαρακτηριστική εικόνα του ψηφιακού κόσμου. Με πάτημα οποιουδήποτε πλήκτρου σ' αυτήν την οθόνη αρχίζει να φορτώνεται η αποστολή της εφαρμογής, και εμφανίζεται ένα μήνυμα στο κέντρο της οθόνης

που ενημερώνει τον χρήστη για την πρόοδο της φόρτωσης. Έτσι, το αρχείο ***Splash2.gui*** διαμορφώνεται ως εξής:

```
//--- OBJECT WRITE BEGIN ---
new GuiChunkedBitmapCtrl(SplashScreen2) {
   profile = "GuiSplashProfile";
   horizSizing = "width";
   vertSizing = "height";
   position = "0 0";
   extent = "640 480";
   minExtent = "8 8";
   visible = "1";
   helpTag = "0";
   bitmap = "./splash2";
   useVariable = "0";
   tile = "0";
   noCursor=1;
   new GuiInputCtrl(SplashScreenInputCtrl) {
      profile = "GuiInputCtrlProfile";
      position = "0 0";
      extent = "10 10";
   };
   new GuiMLTextCtrl(LoadingText) {
     profile = "ScoreLabelProfile";
     horizSizing = "center";
     vertSizing = "center";
     position = "0 0";
     extent = "100 20";
     minExtent = "8 8";
     visible = "1";
     helpTag = "0";
     text = "";
     allowColorChars = "1";
     maxChars = "255";
   };
};
//--- OBJECT WRITE END ---
```

Παρατηρούμε ότι το αρχείο είναι ίδιο με το Splash1.gui εκτός του ότι στη μεταβλητή bitmap το αρχείο είναι το splash2 (το αρχείο είναι το splash2.png αλλά στο Torque δεν χρειάζεται να συμπεριλάβουμε το file extension εκτός αν υπάρχουν δύο αρχεία με το ίδιο όνομα), και ότι εκτός από το αντικείμενο ***SplashScreenInputCtrl***, το ***SplashScreen2*** περιέχει το αντικείμενο ***LoadingText*** που είναι ***GuiMLTextCtrl***, δηλαδή κείμενο πολλαπλών γραμμών. Το profile του αντικειμένου ***LoadingText*** είναι ***ScoreLabelProfile*** και εξηγείται στο σκέλος 4.2.

Μια ακόμη ιδιαιτερότητα είναι ότι ο Listener έχει το ίδιο όνομα με αυτόν της πρώτης οθόνης. Αυτό σημαίνει ότι με το πάτημα κάποιου πλήκτρου καλείται η συνάρτηση ***SplashScreenInputCtrl::onInputEvent*** και στην πρώτη οθόνη και στη δεύτερη, και πρέπει να χρησιμοποιηθεί κάποια τεχνική για να αναγνωριστεί σε ποια οθόνη βρισκόμαστε. Γι' αυτό και στο αρχείο client.cs υπάρχει ο ορισμός της συνάρτησης αυτής ως εξής:

```
function SplashScreenInputCtrl::onInputEvent(%this, %dev, %evt, %make){
  if(%make){
    if(Canvas.getContent() == SplashScreen1.getID()){
      Canvas.setContent(SplashScreen2);
    } else {
      LoadingText.setText("Loading...");
      Schedule(100, 0, "LaunchGame");
      //LaunchGame();
    }
  }
}
```

Η παραπάνω συνάρτηση χειρίζεται οποιοδήποτε πάτημα πλήκτρου όταν είναι ενεργό το **SplashScreenInputCtrl** (δηλαδή στις 2 SplashScreens). Ξεκινά ελέγχοντας το περιεχόμενο του στοιχείου Canvas: αν είναι το SplashScreen1 τότε αλλάζει το content του σε SplashScreen2, αλλιώς γράφει "Loading..." στο LoadingText της SplashScreen2 και ξεκινά την εφαρμογή με την εντολή LaunchGame.

## 4.2: Γραφικό Περιβάλλον

Η εφαρμογή μας απαιτεί την ύπαρξη γραφικού περιβάλλοντος μέσω του οποίου θα εμφανίζονται μηνύματα στο χρήστη που θα τον καθοδηγούν στο ψηφιακό μουσείο. Έτσι, κατά τη διάρκεια της εφαρμογής χρησιμοποιείται το αρχείο ***playerInterface.gui*** που περιέχει όλα τα απαραίτητα αντικείμενα του περιβάλλοντος της εφαρμογής. Τα περιεχόμενα του παρατίθενται παρακάτω:

```
//--- OBJECT WRITE BEGIN ---
new GameTSCtrl(PlayerInterface) {
   profile = "GuiContentProfile";
   horizSizing = "right";
   vertSizing = "bottom";
   position = "0 0";
   extent = "640 480";
   minExtent = "8 8";
   visible = "1";
   helpTag = "0";
   noCursor = "1";

   new GuiChunkedBitmapCtrl(PaintingCover) {
     profile = "GuiDefaultProfile";
     horizSizing = "width";
     vertSizing = "height";
     position = "0 0";
     extent = "640 480";
     minExtent = "8 8";
     visible = "0";
     helpTag = "0";
     bitmap = "";
     useVariable = "0";
     tile = "0";
   };
   new GuiCrossHairHud(GunSight) {
     profile = "GuiDefaultProfile";
     horizSizing = "center";
     vertSizing = "center";
     position = "310 230";
     extent = "20 20";
     minExtent = "20 20";
     visible = "1";
     helpTag = "0";
     bitmap = "./pointer";
     wrap = "0";
   };
   new GuiMLTextCtrl(ScoreLabel) {
     profile = "ScoreLabelProfile";
     horizSizing = "center";
     vertSizing = "center";
     position = "0 0";
     extent = "128 32";
     minExtent = "8 8";
     visible = "0";
     helpTag = "0";
     text = "";
     allowColorChars = "1";
     maxChars = "255";
   };
   new GuiButtonCtrl(ArrowUp) {
     profile = "GuiButtonProfile";
```

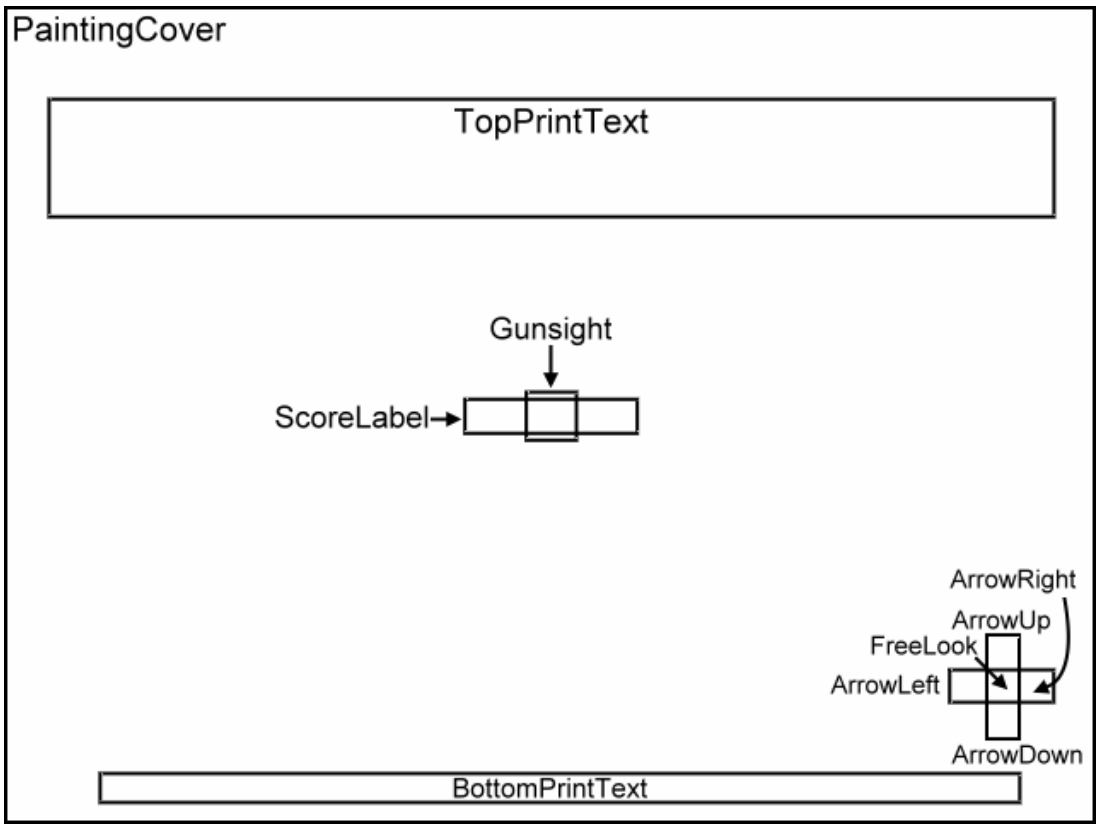
```
      horizSizing = "left";
      vertSizing = "top";
      position = "524 296";
      extent = "24 24";
      minExtent = "24 24";
      visible = "0";
      helpTag = "0";
      text = "U";
      buttonType = "PushButton";
      //bitmap = "./arrow1.gif";
      //command = "Echo(\"ArrowUp clicked\");ArrowUp.setVisible(0);Canvas.cursorOff();";
      command = "Echo(\"ArrowUp clicked\");ClientCameraLook(\"up\");";
      wrap = "0";
   };
    new GuiButtonCtrl(ArrowDown) {
      profile = "GuiButtonProfile";
      horizSizing = "left";
      vertSizing = "top";
      position = "524 344";
      extent = "24 24";
      minExtent = "24 24";
      visible = "0";
      helpTag = "0";
      text = "D";
      buttonType = "PushButton";
      command = "Echo(\"ArrowDown clicked\");ClientCameraLook(\"down\");";
      wrap = "0";
   };
    new GuiButtonCtrl(ArrowLeft) {
      profile = "GuiButtonProfile";
      horizSizing = "left";
      vertSizing = "top";
      position = "500 320";
      extent = "24 24";
      minExtent = "24 24";
      visible = "0";
      helpTag = "0";
      text = "L";
      buttonType = "PushButton";
      command = "Echo(\"ArrowLeft clicked\");ClientCameraLook(\"left\");";
      wrap = "0";
   };
    new GuiButtonCtrl(ArrowRight) {
      profile = "GuiButtonProfile";
      horizSizing = "left";
      vertSizing = "top";
      position = "548 320";
      extent = "24 24";
      minExtent = "24 24";
      visible = "0";
      helpTag = "0";
      text = "R";
      buttonType = "PushButton";
      command = "Echo(\"ArrowRight clicked\");ClientCameraLook(\"right\");";
      wrap = "0";
   };
    new GuiButtonCtrl(FreeLook) {
      profile = "GuiButtonProfile";
      horizSizing = "left";
      vertSizing = "top";
      position = "524 320";
      extent = "24 24";
      minExtent = "24 24";
      visible = "0";
      helpTag = "0";
      text = "FL";
      buttonType = "PushButton";
      command = "Canvas.cursorOff();";
      wrap = "0";
   };
   new GuiMLTextCtrl(BottomPrintText) {
```

```
    profile = "BottomPrintTextProfile";
    horizSizing = "center";
    vertSizing = "top";
    position = "45 450";
    extent = "550 20";
    minExtent = "8 8";
    visible = "0";
    helpTag = "0";
    lineSpacing = "2";
    allowColorChars = "1";
    maxChars = "-1";
  };
  new GuiMLTextCtrl(TopPrintText) {
    profile = "TopPrintTextProfile";
    horizSizing = "center";
    vertSizing = "top";
    position = "20 -50";
    extent = "600 20";
    minExtent = "8 8";
    visible = "0";
    helpTag = "0";
    lineSpacing = "2";
    allowColorChars = "1";
    maxChars = "-1";
    number = "0";
  };
};
//--- OBJECT WRITE END ---
```

Παρατηρούμε κατ' αρχάς ότι όλα τα αντικείμενα περιέχονται στο ***GameTSCtrl*** με όνομα ***PlayerInterface***. Αυτό καθορίζει το μέγεθος του παραθύρου στο οποίο θα τρέχει η εφαρμογή (640 x 480) και ότι δε θα υπάρχει cursor (καθώς η κίνηση του ποντικιού θα ισοδυναμεί με κίνηση του κεφαλιού του avatar).

Τα αντικείμενα που περιέχει το PlayerInterface είναι κατά σειρά:

- ***Painting Cover*** που καλύπτει όλο το γραφικό περιβάλλον όταν θέλουμε να κάνουμε zoom σε ένα πίνακα ή άλλα αντικείμενα.
- ***GunSight*** που σηματοδοτεί στο κέντρο της οθόνης για να έχει ο χρήστης μεγαλύτερο έλεγχο και καλύτερη εικόνα του τι εξετάζει. Η εικόνα που τοποθετείται στο στοιχείο GunSight εξαρτάται από το αντικείμενο στο inventory του χρήστη, όλες οι εικόνες όμως είναι διαφανή PNG.
- ***ScoreLabel*** που βρίσκεται κι αυτό στο κέντρο της οθόνης και παίρνει τη θέση του GunSight όταν ο χρήστης εξετάζει ένα αντικείμενο.
- ***BottomPrintText*** που παρέχει οδηγίες στο χρήστη για τα πλήκτρα που πρέπει να πατήσει κάθε φορά για να αλληλεπιδράσει με τα αντικείμενα της εφαρμογής.
- ***TopPrintText*** που χρησιμοποιείται όπως το BottomPrintText για να ενημερώνει το χρήστη, αλλά σε διαφορετικές περιπτώσεις (π.χ. σε διάλογο με έναν NPC).
- ***ArrowUp, ArrowDown, ArrowRight, ArrowLeft, FreeLook*** είναι πλήκτρα που εμφανίζονται μόνο όταν ο χρήστης εξετάζει ένα έκθεμα του μουσείου και χρησιμεύουν στην κίνηση της κάμερας.




**ΕΙΚΟΝΑ 4:** ΤΟ PLAYERINTERFACE

Παρατηρούμε ότι μεταξύ των άλλων χαρακτηριστικών των αντικειμένων υπάρχει και η μεταβλητή profile που παίρνει τρεις τιμές συνολικά: ***TopPrintProfile, ScoreLabelProfile*** και ***BottomPrintProfile***. Αυτές οι τιμές είναι κάποια έτοιμα πακέτα με πληροφορίες για τις γραμματοσειρές του αντίστοιχου αντικειμένου. Οι πληροφορίες αυτές βρίσκονται στο ***interface_profile.cs*** και είναι τα εξής:

```
new GuiControlProfile ("ScoreLabelProfile"){
  opaque = false;
  fontColor = "255 255 255";
  fontType = "Grave Digger";
  fontSize = 24;
};

// Bottom print
new GuiControlProfile ("BottomPrintTextProfile"){
  opaque = false;
  fontColor = "255 255 255";
  fontType = "Grave Digger";
  fontSize = 20;
};

// Top print
new GuiControlProfile ("TopPrintTextProfile"){
  opaque = false;
  fontColor = "255 255 60";
  fontType = "Bookman Old Style Bold";
  fontSize = 22;
};
```

Όπως βλέπουμε οι πληροφορίες είναι για τις γραμματοσειρές του κειμένου: το χρώμα, το μέγεθος, το είδος και τη διαφάνεια.

## 4.3: Αντιστοίχηση Πλήκτρων

Προφανώς για να μπορεί ο χρήστης να περιηγηθεί μέσα στην εφαρμογή και να αλληλεπιδράσει με τα διάφορα αντικείμενα της, πρέπει να υπάρχει ένα αρχείο που να ορίζει τα πλήκτρα της εφαρμογής και τι κάνουν. Για το σκοπό αυτό υπάρχει το αρχείο keymove.cs τα περιεχόμενα του οποίου είναι:

```
if ( IsObject( playerKeymap ) ){            // If we already have a player key map,
  playerKeymap.delete();                    // delete it so that we can make a new one
}
new ActionMap(playerKeymap);

function DoQuitYesNo(){
  MessageBoxYesNo( "Quit Window", "Exit Museum Walkthrough?", "quit();","");
}

// +-------------------------------------------------------------------------+
// |                     Camera Motion Functions                             |
// +-------------------------------------------------------------------------+

function ClientCameraLook(%direction){
  commandToServer('CameraLook', %direction);
}

// +-------------------------------------------------------------------------+
// |                          Motion Functions                               |
// +-------------------------------------------------------------------------+

function GoLeft(%val){                      // strafing
  $mvLeftAction = %val;
}
function GoRight(%val){
  $mvRightAction = %val;
}
function GoAhead(%val){                     // moving forward/backward
  $mvForwardAction = %val;
}
function BackUp(%val){
  $mvBackwardAction = %val;
}
function DoYaw(%val){                       // looking around
  $mvYaw += %val * ($cameraFov / 90) * 0.01;
}
function DoPitch(%val){
  $mvPitch += %val * ($cameraFov / 90) * 0.01;
}
function DoActivate(%val){                  // activating an item
  if (%val){
    commandToServer('Activate');
  }
}
function DoQuitOrbit(%val){                 // quitting from activation
  if (%val){
    commandToServer('QuitOrbit');
  }
}

function DoCreateBots(%val){                // creating bots
  if (%val){
    commandToServer('CreateBots');
  }
}

function DoDropItem(%val){                  // drop an item from the inventory
  if (%val){
    commandToServer('DropItem');
  }
}
```

```
// +-------------------------------------------------------------------+
// |                           Keymappings                             |
// +-------------------------------------------------------------------+

playerKeymap.Bind(keyboard, "w", GoAhead);
playerKeymap.Bind(keyboard, "s", BackUp);
playerKeymap.Bind(keyboard, "a", GoLeft);
playerKeymap.Bind(keyboard, "d", GoRight);
playerKeymap.Bind(mouse, xaxis, DoYaw);
playerKeymap.Bind(mouse, yaxis, DoPitch);
GlobalActionMap.Bind(keyboard, escape, DoQuitYesNo);
GlobalActionMap.Bind(keyboard, tilde, ToggleConsole);
playerKeymap.Bind(mouse0, "button0", DoActivate);
playerKeymap.Bind(mouse0, "button1", DoDropItem);
playerKeymap.Bind(keyboard, "q", DoQuitOrbit);
playerKeymap.Bind(keyboard, "b", DoCreateBots);
```

Ξεκινάμε από το τέλος, όπου με χρήση της playerKeymap.Bind ορίζονται τα πλήκτρα και οι συναρτήσεις που καλούνται με το πάτημά τους. Η GlobalActionMap.Bind διαφέρει από την προηγούμενη στο ότι πάτημα του εν λόγω πλήκτρου θα καλέσει τη συνάρτηση σε οποιαδήποτε φάση της εφαρμογής, ενώ η playerKeymap.Bind μόνο όταν ο χρήστης περιηγείται στο μουσείο. Παρατηρούμε ότι όλες οι συναρτήσεις που αναφέρονται σαν τρίτη μεταβλητή της Bind βρίσκονται παραπάνω. Είναι εμφανές ότι οι συναρτήσεις κίνησης (GoAhead, BackUp, GoLeft, GoRight, DoYaw, DoPitch) αλλάζουν απλά μια global μεταβλητή, μέσω της οποίας μετακινείται ο χρήστης μέσα στο μουσείο. Οι DoPitch και DoYaw καλούνται μέσα από κίνηση του ποντικιού στον οριζόντιο και στον κάθετο άξονα αντίστοιχα, και αντιστοιχούν σε κίνηση του κεφαλιού του avatar που παρατηρεί το περιβάλλον. Το escape δημιουργεί ένα παράθυρο με επιλογές Yes/No που ενημερώνει τον χρήστη ότι θα βγει από την εφαρμογή. Αν πατηθεί Yes τότε η εφαρμογή τερματίζεται. Το tilde (`) καλεί την κονσόλα, όπου ο χρήστης μπορεί να γράψει εντολές σε κώδικα και να διαβάσει το log της εφαρμογής. Το κουμπί Q χρησιμοποιείται μόνο όταν ο χρήστης ελέγχει την κάμερα αντί για το avatar (π.χ. όταν εξετάζει ένα έκθεμα στο μουσείο) και στέλνει εντολή στον εξυπηρετητή να επιστρέψει στον χρήστη τον έλεγχο του avatar. Το κουμπί B στέλνει εντολή στον εξυπηρετητή να επαναφέρει τους διάφορους NPCs της εφαρμογής (ουσιαστικά χρησιμοποιείται μόνο για κάποιους ελέγχους από τον κατασκευαστή). Τέλος το πάτημα του αριστερού κουμπιού του ποντικιού στέλνει μήνυμα στον εξυπηρετητή να ενεργοποιήσει το αντικείμενο που κοιτάζει ο χρήστης (θα το εξηγήσουμε αυτό εκτεταμένα στο σκέλος 4.4) ενώ με το δεξί κουμπί του ποντικιού στέλνεται μήνυμα στον εξυπηρετητή να ελευθερώσει το inventory του χρήστη σε περίπτωση που αυτός έχει πάρει ένα αντικείμενο.

## 4.4: Εντολές στον Εξυπηρετητή

Για να επικοινωνεί ο πελάτης με τον εξυπηρετητή χρησιμοποιούνται εντολές commandToServer με όρισμα τον τίτλο της συνάρτησης που θέλει ο πελάτης να κληθεί από τον εξυπηρετητή. Από την πλευρά του, ο εξυπηρετητής έχει κάποιους χειριστές αυτών των εντολών μέσω των οποίων επεξεργάζεται τα αιτήματα του πελάτη. Για παράδειγμα, όταν ο χρήστης ενεργοποιήσει ένα αντικείμενο, καλεί την commandToServer('Activate') η οποία στέλνει μήνυμα στον εξυπηρετητή. Ο εξυπηρετητής δέχεται την εντολή αυτή μέσω της συνάρτησης ServerCmdActivate που δέχεται σαν όρισμα τον αριθμό του πελάτη που την καλεί. Η συνάρτηση αυτή επεξεργάζεται το αίτημα ενεργοποίησης του χρήστη και αν χρειαστεί επιστρέφει στον πελάτη ένα μήνυμα με την αντίστοιχη εντολή commandToClient. Το αρχείο που περιέχει όλες τις εντολές στον εξυπηρετητή είναι το sc_command.cs στον φάκελο control/server και περιέχει τις εξής συναρτήσεις:

```
function ServerCmdLook(%client){
  %player = %client.player;
  %eye = %player.getEyeVector();
  %vec = vectorScale(%eye, 3);
  %start = %player.getEyeTransform();
  %end = VectorAdd(%start, %vec);
  %found = ContainerRayCast(%start, %end, ($TypeMasks::StaticShapeObjectType |
$TypeMasks::PlayerObjectType), %player);
  if(%found){
    if(%found.getType() & $TypeMasks::StaticShapeObjectType ){
      %obj_name = %found.name;
      %obj_category = %found.getDatablock().category;
      // **** items for display -- observer mode
      switch$ (%obj_category) {
        case "Display_Item":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          %conn = %client.getControlObject();
          if(%conn == %client.player){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to enter Observer Mode" );
           commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
           commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
           commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
          }
          if(%conn == %client.camera){                                          //
Orbit mode
            commandToClient( %client, 'setGUItext', "BottomPrintText", "Press \c1q\cr to
return to game" );
           commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
           commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
          }
        case "Painting":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          %conn = %client.getControlObject();
          if(%conn == %client.player){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to look at the painting" );
           commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
           commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
           commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
          }
          if(%conn == %client.camera){                                          //
Orbit mode
            commandToClient( %client, 'setGUItext', "BottomPrintText", "Press \c1q\cr to
return to game" );
           commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
           commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
          }
        case "Book":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          %conn = %client.getControlObject();
          if(%conn == %client.player){
```

```
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to read the book" );
            commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
            commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
            commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
          }
          if(%conn == %client.camera){                                          //
Orbit mode
            commandToClient( %client, 'setGUItext', "BottomPrintText", "Press \c1q\cr to
return to game" );
            commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
            commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
          }
        case "Door":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          if( (%found.state $= "closed") || (%found.state $= "locked") ){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to open the door" );
          } else if(%found.state $= "open"){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to close the door" );
          } else {
            commandToClient( %client, 'setGUItext', "BottomPrintText", "" );
          }
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
          commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
        case "Chest":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          if( %found.getDataBlock().getName() $= "DB_chest_base" ){
            %obj_lid = %found.getName() @ "_lid";
            %found = %obj_lid;
          }
          if( (%found.state $= "closed") || (%found.state $= "locked") ){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to open the chest" );
          } else if(%found.state $= "open"){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to close the chest" );
          } else {
            commandToClient( %client, 'setGUItext', "BottomPrintText", "" );
          }
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
          commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
        case "Desk":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          if( (%found.state $= "closed") || (%found.state $= "locked") ){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to open the desk drawer" );
          } else if(%found.state $= "open"){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to close the desk drawer" );
          } else {
            commandToClient( %client, 'setGUItext', "BottomPrintText", "" );
          }
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
          commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
        case "Lamp":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          if(%found.state $= "lit"){
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to put out the torch" );
          } else {
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to light the torch" );
          }
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
          commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
```

```
        case "Key":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr to
pick up the " @ %obj_name );
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
          commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
        case "Desk_Item":
          commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
          %conn = %client.getControlObject();
          if(%conn == %client.player){
            commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
            commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr
to use the " @ %obj_name );
            commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
            commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
          }
          if(%conn == %client.camera){                                          //
Orbit mode
            commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
            commandToClient( %client, 'setGUItext', "BottomPrintText", "Press \c1q\cr to
return to game" );
            commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
          }
        // END OF SWITCH
      }
    }
    // **** bots (NPCs) -- can be activated
    if(%found.getType() & $TypeMasks::PlayerObjectType){
      %obj_name = %found.getDatablock().getName();
      commandToClient( %client, 'setGUIvisible', "GunSight", 0 );
      commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
      commandToClient( %client, 'setGUItext', "BottomPrintText", "\c1Left-Click\cr to
activate bot" );
      commandToClient( %client, 'setGUIvisible', "ScoreLabel", 1 );
      commandToClient( %client, 'setGUItext', "ScoreLabel", %obj_name );
    }
  } else {
    commandToClient( %client, 'setGUIvisible', "GunSight", 1 );
    commandToClient( %client, 'setGUIvisible', "BottomPrintText", 0 );
    commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
  }
}
```

Η παραπάνω εντολή είναι ουσιαστικά η εντολή Look, μέσω της οποίας ο χρήστης επεξεργάζεται το περιβάλλον του, και σε περίπτωση που κοιτάζει μέσα από τα μάτια του avatar ένα αλληλεπιδραστικό αντικείμενο προβάλλονται βοηθητικά μηνύματα στην οθόνη με το όνομα του αντικειμένου (στο ScoreLabel) και τον τρόπο που μπορεί ο χρήστη να το ενεργοποιήσει (στο BottomPrintText). Η εντολή ξεκινά βρίσκοντας τον avatar που χειρίζεται ο πελάτης που την κάλεσε και το διάνυσμα που αντιστοιχεί στο «βλέμμα» του. Το EyeVector είναι ένα μοναδιαίο διάνυσμα που ξεκινά από τα μάτια του μοντέλου του avatar και αντιστοιχεί στο «βλέμμα» του avatar. Επειδή το EyeVector είναι μοναδιαίο, το πολλαπλασιάζουμε με το 3 έτσι ώστε ο χρήστης να μπορεί να εξετάζει αντικείμενα χωρίς να χρειάζεται να τα πλησιάσει πολύ. Για να υπολογίσουμε το αντικείμενο που εξετάζει ο χρήστης πρέπει να προσθέσουμε το τροποποιημένο EyeVector στη θέση των ματιών στο μοντέλο του avatar (με χρήση της εντολής getEyeTransform). Με χρήση της εντολής ContainerRayCast βρίσκουμε το πρώτο αλληλεπιδραστικό αντικείμενο που βρίσκεται μεταξύ της θέσης των ματιών του avatar και του τέλους του EyeVector. Αν βρεθεί πράγματι ένα τέτοιο αντικείμενο, τότε θα είναι είτε StaticShape είτε PlayerObject (γιατί αυτά τα δύο είδη αντικειμένων είχαμε θέσει σαν «μάσκες» στην ContainerRayCast) και σώζεται στη μεταβλητή found. Κατόπιν ελέγχεται ο τύπος της found: αν είναι StaticShape (δηλαδή στατικό αλληλεπιδραστικό αντικείμενο) τότε ελέγχεται και η κατηγορία του dataBlock του. Ανάλογα με την κατηγορία αυτή αλλάζει και το κείμενο στο BottomPrintText. Ανεξάρτητα από τον τύπο και την κατηγορία, το GunSight εξαφανίζεται και τη θέση του παίρνει το

ScoreLabel στο οποίο αναγράφεται το όνομα του αντικειμένου. Τέλος, αν η ContainterRayCast δεν βρει αντικείμενο τότε εξαφανίζονται τα ScoreLabel και BottomPrintText και εμφανίζεται το GunSight.

```
function ServerCmdActivate(%client){                                          // Examining a
Display Item

  %player = %client.player;
  Echo( "Player Position: " @ %player.getTransform() );

  %player = %client.player;
  %eye = %player.getEyeVector();
  %vec = vectorScale(%eye, 3);
  %start = %player.getEyeTransform();
  %end = VectorAdd(%start, %vec);
  %found = ContainerRayCast(%start, %end, ($TypeMasks::StaticShapeObjectType |
$TypeMasks::PlayerObjectType), %player);
  if(%found){
    if(%found.getType() & $TypeMasks::StaticShapeObjectType){
      %obj_name = %found.name;
      %obj_category = %found.getDatablock().category;
      // **** items for display -- observer mode
      switch$ (%obj_category) {
        case "Display_Item":
          if( getWord(Showbot1.status, 0) $= "diorama" ){
            Showbot1.talk_i = %found.talk_id;                                 // BOT
ADDITION
          }
          Echo("Observing " @ %obj_name @ " of class " @ %obj_category @ "!");
         %found.getDatablock().freeLook(%found, %client);
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
          //latest additions
          commandToClient( %client, 'setGUIvisible', "ArrowSet", 1 );
          commandToClient( %client, 'activateCursor', 1 );
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUItext', "BottomPrintText", "Press \c1q\cr to
exit Observer Mode" );
        case "Painting":
          Error("Looking at painting " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
          commandToClient( %client, 'setGUIvisible', "PaintingCover", 1 );
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUItext', "BottomPrintText", "Press \c1q\cr to
return to game" );
        case "Book":
          Error("Reading book " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
          commandToClient( %client, 'setGUIvisible', "PaintingCover", 1 );
          commandToClient( %client, 'setGUIvisible', "ScoreLabel", 0 );
          commandToClient( %client, 'setGUIvisible', "BottomPrintText", 1 );
          commandToClient( %client, 'setGUItext', "BottomPrintText", "Press \c1q\cr to
return to game" );
        case "Door":
          Error("Activating door " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
          if( ( %found.getID() $= door_gallery.getID() ) && ( door_gallery.state $=
"locked" ) ){
            Error( "Alert Showbot!" );
            if( Showbot1.status $= "bot follow" ){
              Showbot1.talk_i = 1;
              Showbot1.talk_j = 0;
              Showbot1.status = "gallery talk";
            }
          }
        case "Lamp":
          Error("Using lamp " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
        case "Chest":
          if( %found.getDataBlock().getName() $= "DB_chest_base" ){
```

```
            %obj_lid = %found.getName() @ "_lid";
            %found = %obj_lid;
          }
          Error("Opening Chest " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
        case "Desk":
          Error("Opening Desk Drawer " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
        case "Desk_Item":
          Error("Using/Picking up item " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
        case "Key":
          Error("Using/Picking up item " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
        case "Magic_Button":
          Error("Using button " @ %found.getID() @ "!");
          %found.getDatablock().activate(%found, %client);
        // END OF SWITCH
      }
      return;
    }
    // **** bots (NPCs) -- can be activated
    if(%found.getType() & $TypeMasks::PlayerObjectType){
      Error("Activating bot " @ %found.getID() @ "!");
      %found.getDatablock().activate(%found, %client);
      return;
    }
  }
}
```

Η παραπάνω εντολή είναι ουσιαστικά η εντολή Activate με την οποία ο χρήστης ενεργοποιεί τα αλληλεπιδραστικά αντικείμενα της εφαρμογής. Η συνάρτηση ξεκινά με τον ίδιο τρόπο όπως η Look, δηλαδή με EyeVector, ContainerRayCast κτλ. Και πάλι λοιπόν ελέγχεται ο τύπος του found και αν είναι staticShape ελέγχεται και η κατηγορία του. Στην περίπτωση που το αντικείμενο έχει κάποια εικόνα που θα εμφανιστεί σε όλη την οθόνη (π.χ. ένα βιβλίο ή ένας πίνακας) τότε εξαφανίζονται όλα τα υπόλοιπα αντικείμενα του γραφικού περιβάλλοντος και εμφανίζεται σε όλη την οθόνη το PaintingCover. Στην περίπτωση που ενεργοποιούμε ένα έκθεμα, τότε ο NPC ξεναγός ενημερώνεται ως προς το ποιο έκθεμα εξετάζεται για να μας ενημερώσει γι' αυτό. Τα υπόλοιπα αντικείμενα απλά χρησιμοποιούν την εντολή activate του DataBlock που τους αντιστοιχεί.

```
function ServerCmdQuitOrbit(%client){
  %conn = %client.getControlObject();
  if (%conn == %client.camera){
    %conn.mode = "Follow";
    Echo("Quitting orbit from camera " @ %client.camera @ "!");
    %transform = %client.player.getTransform();
    %conn = %client.player;
    %client.camera.setTransform(%conn.getEyeTransform());
    %client.camera.setVelocity("0 0 0");
    %client.setControlObject(%conn);
    commandToClient( %client, 'setGUIvisible', "PaintingCover", 0 );
    commandToClient( %client, 'setGUIvisible', "ArrowSet", 0 );
    commandToClient( %client, 'activateCursor', 0 );
    if( getWord(Showbot1.status, 0) $= "diorama" ){
      ShowBot1.getDatablock().activate( Showbot1.getID(), %client );
    }
  }
}
```

Η εντολή QuitOrbit χρησιμοποιείται όταν ο χρήστης θέλει να ανακτήσει τον έλεγχο του avatar στις περιπτώσεις που εξετάζει ένα αντικείμενο και ελέγχει την κάμερα. Η συνάρτηση αυτή τοποθετεί την κάμερα στο σημείο των ματιών του avatar και επιστρέφει τον έλεγχο του avatar στον χρήστη. Επίσης επαναφέρει όλα τα στοιχεία του γραφικού περιβάλλοντος που

είχαν εξαφανιστεί κατά τη διάρκεια της εξέτασης του αντικειμένου. Τέλος ο NPC οδηγός ενεργοποιείται για να μπορέσει να ενημερώσει τον χρήστη για το αντικείμενο που εξέταζε.

```
function ServerCmdCreateBots(%client){
  CC_Bots();
}
```

Η εντολή αυτή καλείται αν ο χρήστης θέλει να επαναφέρει τους NPCs της εφαρμογής, και καλεί την εντολή CC_Bots που βρίσκεται στο bot.cs το οποίο θα εξετάσουμε σε ακόλουθα κεφάλαια.

```
function ServerCmdCameraLook(%client, %direction){
  %conn = %client.getControlObject();
  Echo("Moving " @ %direction @ " the camera " @ %conn.target);
  %conn.target.getDatablock().Look(%client, %direction);
}
```

Η εντολή αυτή καλείται μόνο όταν εξετάζεται ένα έκθεμα και ο χρήστης πατήσει ένα από τα κουμπιά κατεύθυνσης της κάμερας.

```
function ServerCmdDropItem(%client){
  if( %client.inventory ){
    %client.inventory.setScale( %client.inventoryXscale );
    %client.inventory = 0;
    commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
  }
}
```

Η εντολή αυτή τοποθετεί το αντικείμενο που έχει ο χρήστης στο inventory του στο σημείο που βρισκόταν πριν, και επαναφέρει το στοιχείο GunSight του GUI.

## 4.5: Εντολές στον Πελάτη

Ο εξυπηρετητής έχει όλες τις πληροφορίες για το πώς λειτουργούν τα διάφορα αντικείμενα στην εφαρμογή, αλλά και πως αλληλεπιδρά ο χρήστης με αυτά. Παρ' όλ' αυτά, τα μηνύματα που εμφανίζονται στην οθόνη του χρήστη δεν είναι αρμοδιότητα του εξυπηρετητή: ο εξυπηρετητής στέλνει μήνυμα στον πελάτη (με το commandToClient) με εντολή να τροποποιήσει κάποιο στοιχείο του γραφικού περιβάλλοντος. Το αρχείο που περιέχει όλες τις εντολές προς τον πελάτη είναι το cs_command.cs στο φάκελο control/client και περιέχει τις εξής συναρτήσεις:

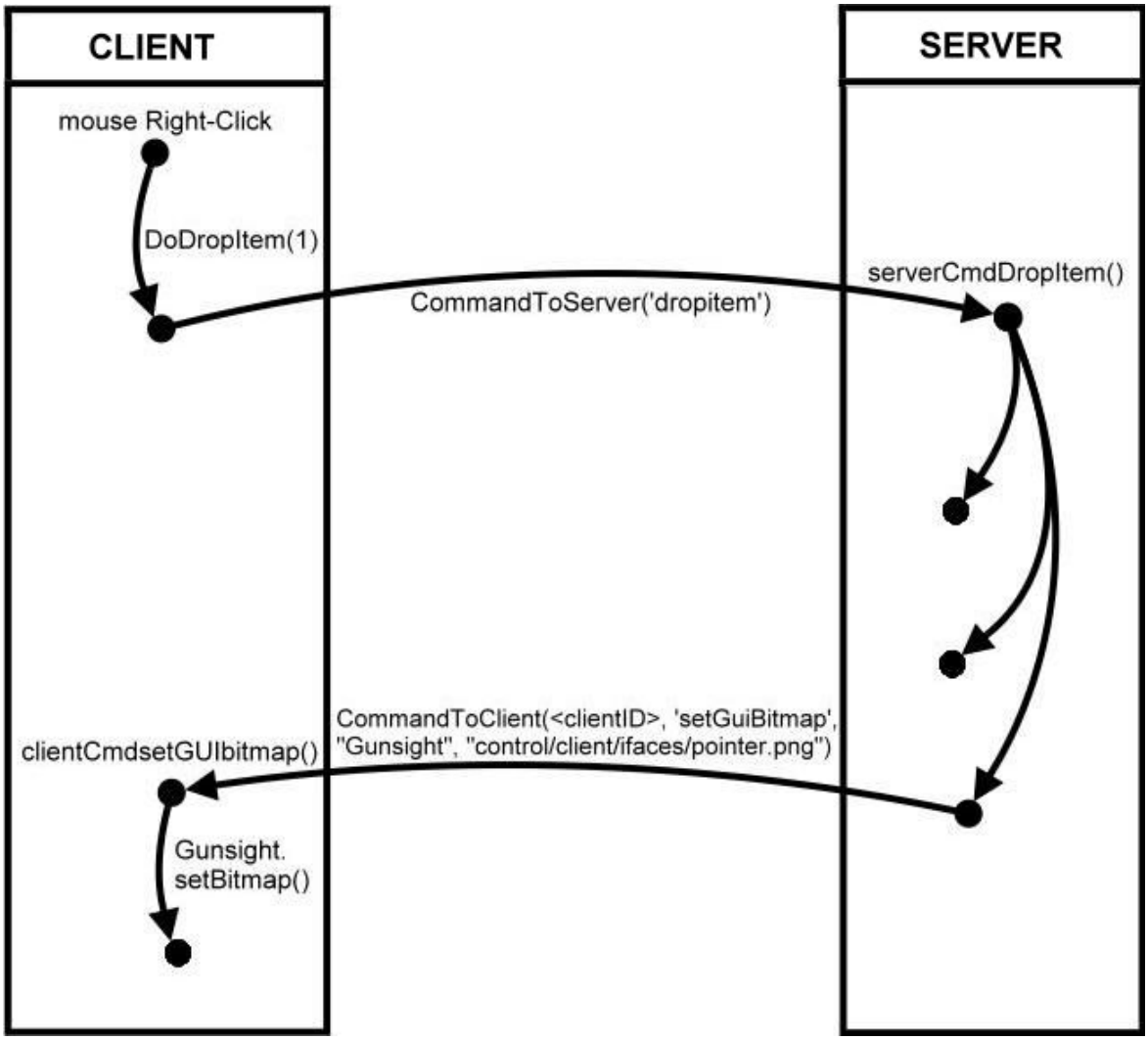
```
function clientCmdsetGUIvisible( %gui, %value ){
  switch$ ( %gui ){
    case "GunSight":
      GunSight.setVisible(%value);
    case "ScoreLabel":
      ScoreLabel.setVisible(%value);
    case "BottomPrintText":
      BottomPrintText.setVisible(%value);
    case "ArrowSet":
      ArrowUp.setVisible(%value);
      ArrowDown.setVisible(%value);
      ArrowLeft.setVisible(%value);
      ArrowRight.setVisible(%value);
      FreeLook.setVisible(%value);
    case "PaintingCover":
      PaintingCover.setVisible(%value);
  }
}
```

```
function clientCmdsetGUItext( %gui, %text ){
  switch$ ( %gui ){
    case "ScoreLabel":
      ScoreLabel.setText("<just:center>" @ %text);
    case "BottomPrintText":
      BottomPrintText.setText("<just:center>" @ %text);
  }
}

function clientCmdsetGUIbitmap( %gui, %filename ){
  switch$ ( %gui ){
    case "GunSight":
      Gunsight.setBitmap(%filename);
    case "PaintingCover":
      PaintingCover.setBitmap(%filename);
  }
}

function clientCmdactivateCursor( %value ){
  if( %value == 1 ){
    Canvas.setCursor("DefaultCursor");
    Canvas.cursorOn();
  } else {
    Canvas.cursorOff();
  }
}

function clientCmdwriteHint( %text ){
  WriteHint(%text);
}
```

Η εντολή SetGUIVisible εμφανίζει ή εξαφανίζει (αναλόγως της τιμή του Boolean ορίσματος value) το στοιχείο του γραφικού περιβάλλοντος που εξαρτάται από το όρισμα gui. Όμοια, η εντολή SetGUIText και SetGUIBitmap τοποθετούν ένα κείμενο ή μια εικόνα αντίστοιχα στα στοιχεία του περιβάλλοντος που εξαρτώνται από το όρισμα gui. Η εντολή ActivateCursor εμφανίζει τον cursor αν το όρισμα value της εντολής είναι 1. Η ίδια εντολή εξαφανίζει τον cursor, οπότε ο «προσανατολισμός» του avatar γίνεται με χρήση του ποντικιού. Τέλος, η εντολή WriteHint καλεί τη συνάρτηση WriteHint από την πλευρά του πελάτη, η οποία βρίσκεται στο αρχείο hintbox που ακολουθεί:

```
function WriteHint(%text){
  Echo("Writing hint: " @ %text );
  %number = GetWord(TopPrintText.number,0);
  %number = %number + 1;
  TopPrintText.number = "" @ %number @ "";
  TopPrintText.setText("<just:center>" @ %text );
  TopPrintText.setVisible(1);
  Schedule(4000, 0, "FadeOutHint", %number);
}

function FadeOutHint(%number){
  %current_number = GetWord(TopPrintText.number,0);
  if( %number == %current_number ){
    TopPrintText.setText("");
    TopPrintText.setVisible(0);
  }
}
```

Η εντολή WriteHint τοποθετεί ένα κείμενο στο στοιχείο TopPrintText και όταν περάσουν 4 δευτερόλεπτα το εξαφανίζει με χρήση της FadeOutHint. Επειδή μπορεί κατά τη διάρκεια της αναμονής (των 4 δευτερολέπτων) να έχει ξανακληθεί η WriteHint, προστίθεται μια μεταβλητή number στο αντικείμενο TopPrintText, και όταν καλείται η FadeOutHint εξαφανίζει το TopPrintText μόνο αν οι αριθμοί συμπίπτουν. Έτσι εξασφαλίζεται ότι δε θα εξαφανιστεί το TopPrintText πριν προλάβει ο χρήστης να το διαβάσει.




**ΕΙΚΟΝΑ 5:** ΠΑΡΑΔΕΙΓΜΑ ΛΕΙΤΟΥΡΓΙΑΣ ΤΗΣ CLIENT/SERVER ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΣΤΗΝ ΕΦΑΡΜΟΓΗ

# ΚΕΦΑΛΑΙΟ 5:
# Κατασκευή του εικονικού κόσμου

## 5.1: Η έννοιες του αντικειμένου και του DataBlock

Τα αντικείμενα (objects) είναι στιγμιότυπα κλάσεων αντικειμένων (object classes), οι οποίες αποτελούν μια συλλογή ιδιοτήτων και μεθόδων που καθορίζουν ένα συγκεκριμένο σύνολο συμπεριφορών και χαρακτηριστικών. Μετά από τη δημιουργία του, ένα αντικείμενο παίρνει ένα μοναδικό αριθμό που αποκαλείται handle. Όταν δύο handles έχουν την ίδια τιμή, αναφέρονται στο ίδιο αντικείμενο. Όταν ένα αντικείμενο υπάρχει σε μια εφαρμογή πολλαπλών χρηστών με έναν κεντρικό εξυπηρετητή και πολλούς πελάτες, ο εξυπηρετητής και κάθε πελάτης διαθέτει το handle του για την αποθήκευση του αντικειμένου στη μνήμη. Να σημειωθεί ότι τα dataBlocks (ένα ειδικό είδος αντικειμένου) αντιμετωπίζονται διαφορετικά.

### 5.1.1: Δημιουργία ενός αντικειμένου

Όταν δημιουργούμε το στιγμιότυπο ενός αντικειμένου, μπορούν να αρχικοποιηθούν πολλές μεταβλητές του αντικειμένου μέσα στον κώδικα new:

Παράδειγμα:

```
%torch_handle = new StaticShape(torch1) {
   position = "-7.9 -4.35 -0.35";
   rotation = "1 0 0 0";
   scale = "1 1 1";
   dataBlock = "DB_torch";
   state = "lit";
   name = "Torch";
};
```

Το handle του αντικειμένου StaticShape που δημιουργείται σώζεται στην μεταβλητή torch_handle. Παρατηρούμε ότι οι παράμετροι του αντικειμένου είναι οι Position (θέση), Rotation (γωνία), Scale (μέγεθος), dataBlock. Αυτές οι παράμετροι είναι απαραίτητες για να εμφανιστεί ένα αντικείμενο StaticShape στην εφαρμογή. Εκτός από αυτές όμως υπάρχουν και δύο «παραπανίσιες» παράμετροι, οι name και state οι οποίες δεν είναι απαραίτητες, αλλά χρησιμοποιούνται από την εφαρμογή για να καθοριστεί το όνομα του αντικειμένου και η κατάσταση του αντίστοιχα. Οι «παραπανίσιες» παράμετροι δεν αναγνωρίζονται από τον κώδικα μηχανής, αλλά μπορούν να διαβαστούν από τα αρχεία TorqueScript της εφαρμογής.

### 5.1.2: Χρήση ενός αντικειμένου

Για να χρησιμοποιήσουμε ένα αντικείμενο, μπορούμε να χρησιμοποιήσουμε το handle του για να αποκτήσουμε πρόσβαση στις παραμέτρους του και τις συναρτήσεις του. Η πρόσβαση σε μία παράμετρο γίνεται με τον εξής τρόπο (για το παραπάνω αντικείμενο):

```
%torch_handle.name = “Torch2”;
```

Όμως τα handles δεν είναι οι μόνοι τρόποι για να αναγνωρίζουμε αντικείμενα: μπορούμε να χρησιμοποιήσουμε το όνομα του αντικειμένου, αρκεί το όνομα αυτό να είναι μοναδικό. Για παράδειγμα, για το παραπάνω αντικείμενο μπορούμε να γράψουμε:

```
Torch1.name = “Torch2”;
```

### 5.1.3: Συναρτήσεις αντικειμένων

Αν το αντικείμενο έχει μια συνάρτηση ορισμένη για το ίδιο, τότε μπορεί να την καλέσει ως εξής:

```
%torch_handle.light( “Arg”, “Arg0” );
```

Ο τρόπος που ορίζεται μια συνάρτηση για ένα συγκεκριμένο αντικείμενο είναι η εξής:

```
Torch1::light( %this, %arg1, %arg2 ){}
```

Ως επί το πλείστον όμως οι συναρτήσεις ορίζονται για κλάσεις και όχι για αντικείμενα, και επειδή οι κλάσεις είναι σύνολα από DataBlocks (όπως θα αναλύσουμε παρακάτω), για να κληθεί μια τέτοια συνάρτηση πρέπει να γραφεί το εξής:

```
%torch_handle.getDataBlock().light( “Arg”, “Arg0” );
```

Όταν καλείται μια συνάρτηση, το πρώτο όρισμα είναι το handle του αντικειμένου που περιέχει την συνάρτηση. Γι’ αυτό και στον παραπάνω ορισμό της συνάρτησης υπάρχει ένα παραπάνω όρισμα, το %this. Το όρισμα %this δεν χρησιμοποιείται όταν καλούμε τη συνάρτηση, αλλά προστίθεται από την μηχανή αυτόματα.

### 5.1.4: DataBlocks

Το DataBlock είναι ένας ειδικός τύπος αντικειμένου που περιέχει ένα σύνολο χαρακτηριστικών που χρησιμοποιούνται για να περιγράψουν τις ιδιότητες ενός άλλου αντικειμένου. Τα DataBlocks υπάρχουν ταυτόχρονα και στον εξυπηρετητή και σε όλους τους πελάτες.
Τα DataBlocks ανήκουν σε κάποιες κατηγορίες (PlayerData, StaticShapeData, ItemData κτλ.), ανάλογα και με το είδος των αντικειμένων που περιγράφουν (Player, StaticShape, Item κτλ.). Κάθε είδος Data έχει διαφορετικές παραμέτρους που μπορούν να οριστούν. Για παράδειγμα, το PlayerData έχει πολλές παραμέτρους για την ταχύτητα του avatar, τη μέγιστη γωνία που μπορεί να ανέβει κτλ. Αντίθετα, στο StaticShapeData αρκεί να οριστεί η κλάση του DataBlock και το αρχείο του μοντέλου που το απεικονίζει. Όταν ένα αντικείμενο «δένεται» σε ένα DataBlock, κληρονομεί όλες τις παραμέτρους του και τις συναρτήσεις του (εκτός αν ορίσει δικές του παραμέτρους και συναρτήσεις, με τα ίδια ονόματα με τις αντίστοιχες του DataBlock).
Παρόλο που τα αντικείμενα μπορεί να δημιουργούνται και να διαγράφονται συνεχώς μέσα σε μια εφαρμογή, τα DataBlocks δημιουργούνται όταν φορτώνεται η αποστολή και δεν διαγράφονται. Ο τρόπος που συντάσσεται το DataBlock είναι ο εξής:

```
datablock StaticShapeData(DB_torch){
  className = "Torch";
  category = "Lamp";
  shapefile = "~/data/active/lamp/torch.dts";
};
```

Παρατηρούμε ότι το DataBlock με όνομα DB_torch είναι StaticShapeData (και άρα τα αντικείμενα που περιγράφει πρέπει να είναι StaticShape), ενώ στο εσωτερικό του ορίζεται η κλάση στην οποία ανήκει και το μοντέλο με το οποίο απεικονίζεται στην εφαρμογή. Επίσης υπάρχει η «παραπανίσια» παράμετρος category, που χρησιμοποιείται για να ομαδοποιούνται παρόμοια DataBlocks στην εφαρμογή.

### 5.1.5: Κλάσεις

Μία κλάση καθορίζει τα αφηρημένα χαρακτηριστικά ενός αντικειμένου και τα πράγματα που μπορεί να κάνει. Παραδείγματος χάριν, η κλάση Torch θα αποτελούταν από τα γνωρίσματα κοινά σε όλους του πυρσούς, π.χ. τον τύπο φωτιάς που παράγουν, ενώ ταυτόχρονα θα όριζε συναρτήσεις (όπως activate ή light) τις οποίες καλεί κάθε μέλος της με τους τρόπους που παρουσιάστηκαν στο σκέλος **5.1.3: Συναρτήσεις αντικειμένων**. Οι κλάσεις παρέχουν την τμηματικότητα και τη δομή σε ένα αντικειμενοστραφές πρόγραμμα.

## 5.2: Το αρχείο της αποστολής

Η πλειονότητα των αντικειμένων της εφαρμογής βρίσκονται στο αρχείο αποστολής (με κατάληξη .mis) το οποίο βρίσκεται στο φάκελο control/data/maps. Τα αντικείμενα της αποστολής που καταχωρήθηκαν στο αρχείο αποστολής είναι είτε TSStatic είτε StaticShape, καθώς και μερικά ειδικά αντικείμενα.

### 5.2.1: Ειδικά αντικείμενα

Τα ειδικά αντικείμενα περιγράφουν όλη την αποστολή και δεν απεικονίζουν κάτι συγκεκριμένο.

```
new ScriptObject(MissionInfo) {
   desc0 = "A basic somewhat smooth map.";
   name = "Museum";
};
new MissionArea(MissionArea) {
   area = "-152 -352 1040 1056";
   flightCeiling = "300";
   flightCeilingRange = "20";
   locked = "true";
};
new Sun(Omnilight) {
   direction = "0.635001 0.635001 -0.439941";
   color = "0.888000 0.885000 0.780000 1.000000";
   ambient = "0.400000 0.400000 0.400000 1.000000";
   locked = "true";
   position = "0 0 0";
   rotation = "1 0 0 0";
   scale = "1 1 1";
};
```

Το ScriptObject απλά περιέχει μια περιγραφή της αποστολής και το όνομά της.
Το MissionArea εκφράζει τα όρια της αποστολής τόσο στην οριζόντια επιφάνεια (area) όσο και στο μέγιστο ύψος (flightCeiling). Τα ουσιαστικά όρια της αποστολής είναι πολύ πιο περιορισμένα, αλλά ο ορισμός του MissionArea είναι απαραίτητος από τον κώδικα μηχανής γι' αυτό και παρατίθεται.
Το Sun είναι κατά μια έννοια ένα αντικείμενο, αλλά δεν αντιπροσωπεύει τόσο τον ήλιο όσο μια γενική πληροφορία για το διάχυτο φως της εφαρμογής. Η κατεύθυνση των ακτινών φωτός καθορίζουν ποιες επιφάνειες θα είναι σκουρότερες και ποιες πιο φωτισμένες. Επίσης υπάρχει πληροφορία για το χρώμα του φωτός αλλά και το χρώμα της «σκιάς» (ambient) δηλαδή πόσο σκουρότερο θα είναι μια επιφάνεια που δεν φωτίζεται.

### 5.2.2: TSStatic

Αντικείμενα που ανήκουν στην κατηγορία TSStatic είναι στατικά και στην εφαρμογή αυτή τα TSStatic αντικείμενα δεν είναι αλληλεπιδραστικό. Αντίθετα με τα StaticShape αντικείμενα που θα αναλύσουμε παρακάτω, τα TSStatic αντικείμενα έχουν μόνο τις απολύτως απαραίτητες παραμέτρους, δηλαδή τη θέση (position), γωνία (rotation), μέγεθος (scale) και μοντέλο (shapeName). Τα TSStatic αντικείμενα δεν έχουν DataBlocks ούτε συναρτήσεις: απλά υπάρχουν στον χώρο της αποστολής. Παραδείγματα TSStatic αντικειμένων στην εφαρμογή είναι τα δωμάτια, η βιβλιοθήκη, το γραφείο (χωρίς τα συρτάρια) και τα παράθυρα.

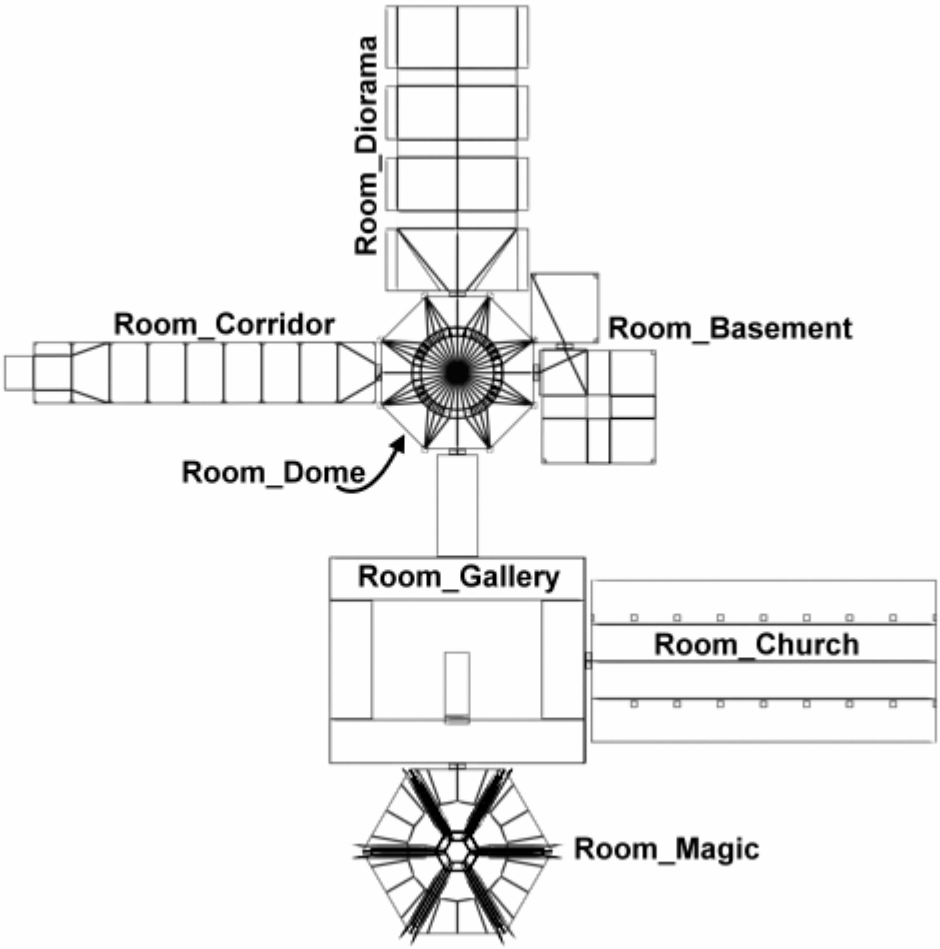
### 5.2.3: StaticShape

Αντικείμενα της κατηγορίας StaticShape στην εφαρμογή ανήκουν πάντα σε ένα DataBlock στο οποίο υπάρχουν πολλές από τις παραμέτρους τους (όπως το μοντέλο 3D που τα απεικονίζει). Τα StaticShapes έχουν όλα ένα όνομα (με την παράμετρο name) το οποίο απεικονίζεται όταν ο avatar εξετάζει ένα αντικείμενο. Επίσης όλα τα StaticShapes είναι αλληλεπιδραστικά, που σημαίνει ότι μπορεί ο avatar να τα χρησιμοποιήσει. Γι' αυτό και όλα τα DataBlocks της κατηγορίας StaticShapeData έχουν συνάρτηση activate (ακόμα κι αν αυτή δεν κάνει τίποτα). Τα StaticShapes έχουν τις απαραίτητες παραμέτρους (θέση, γωνία, μέγεθος, DataBlock) αλλά συχνά περιλαμβάνουν «παραπανίσιες» παραμέτρους που χρησιμοποιεί ο κώδικας για ειδικές χρήσεις.

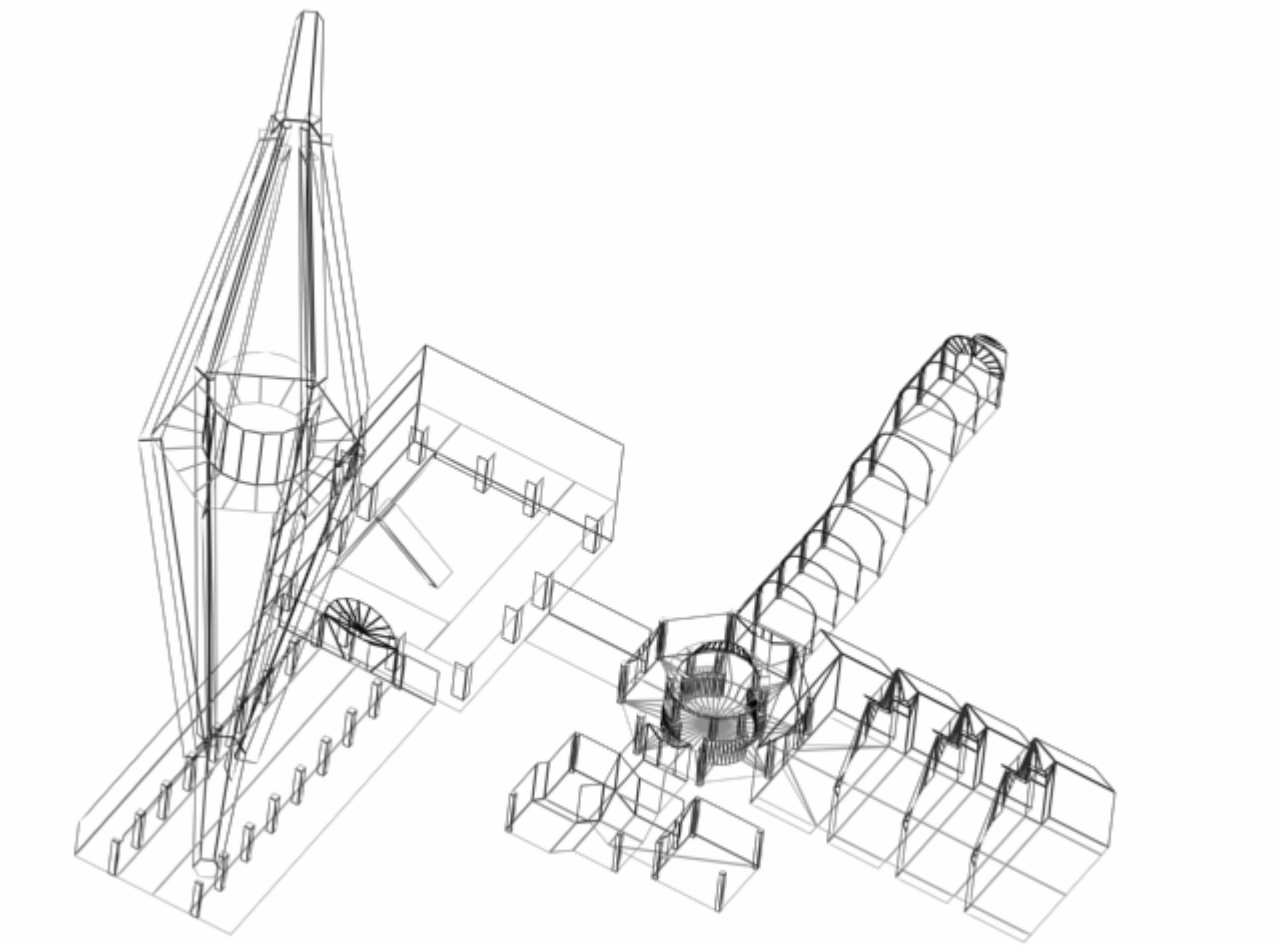
## 5.3: Σύντομη Περιγραφή των χώρων της Εφαρμογής

Ο χάρτης της εφαρμογής περιλαμβάνει τα ακόλουθα δωμάτια:

- **Room_Corridor:** είναι το δωμάτιο που περιλαμβάνει την διπλή πόρτα από την οποία ο χρήστης βγαίνει από το μουσείο (και άρα και την εφαρμογή) και μετά από αρκετές σειρές σκαλοπάτια και διάφορους πυρσούς στους πλαϊνούς τοίχους φτάνει στην πόρτα που τον οδηγεί στο Room_Dome. Ο avatar ξεκινά σ' αυτό το δωμάτιο, μπροστά από την διπλή πόρτα εξόδου.
- **Room_Dome:** είναι το οκταγωνικό δωμάτιο που έχει πόρτες για όλα σχεδόν τα δωμάτια της εφαρμογής. Δεν περιέχει παρά ένα θόλο, ένα φωτιστικό που κρέμεται από το ταβάνι, μερικούς πυρσούς και ένα κλειδί πάνω σε ένα καρφί. Το δωμάτιο έχει 4 πόρτες, 2 από τις οποίες είναι κλειδωμένες κατά την εκκίνηση της εφαρμογής. Οι πόρτες για το room_corridor και το room_staircase είναι ξεκλείδωτες, ενώ οι πόρτες για το room_diorama και το room_gallery είναι κλειδωμένες. Θα ξεκλειδώσουν μόνο όταν δοθεί ένας σφραγισμένος φάκελος με εξουσιοδότηση στον ξεναγό του μουσείου, που βρίσκεται σ' αυτό το δωμάτιο, κάτω από τον θόλο.
- **Room_Diorama:** είναι το δωμάτιο στο οποίο βρίσκονται οι μακέτες του μουσείου. Είναι ένα ορθογώνιο δωμάτιο με ψηλούς κίονες και παράθυρα. Επίσης περιέχει τέσσερις λάμπες δαπέδου, τέσσερα μεγάλα λάβαρα και οκτώ ξύλινες κατασκευές που περιέχουν τις μακέτες που αποτελούν τα εκθέματα αυτού του δωματίου.
- **Room_Basement:** αποτελείται από μια σκάλα και ένα δωμάτιο με κλειδωμένη πόρτα. Η πόρτα ανοίγει με το κλειδί που βρίσκεται στο δωμάτιο Room_Dome. Το δωμάτιο αυτό περιέχει ένα κρεβάτι, μια βιβλιοθήκη, ένα γραφείο και το σκαμνί του. Επίσης υπάρχει ένα κλειδωμένο σεντούκι που ανοίγει με το κλειδί που βρίσκεται κάτω από το μαξιλάρι του κρεβατιού. Η βιβλιοθήκη περιέχει μερικές δεκάδες βιβλία, από τα οποία μόνο ένα χρησιμεύει στην σύνταξη της εξουσιοδότησης που απαιτείται για να ξεκινήσει η ξενάγηση του μουσείου. Στο γραφείο επίσης υπάρχουν διάφορα αντικείμενα που αλληλεπιδρούν για την σύνταξη της εξουσιοδότησης: ένα λευκό χαρτί, πένα φτερού, μελάνι, ένας φάκελος (που βρίσκεται μέσα στο συρτάρι του γραφείου), το δαχτυλίδι με τη σφραγίδα του μουσείο (που βρίσκεται μέσα στο σεντούκι), και κερί για να σφραγιστεί ο φάκελος.
- **Room_Gallery:** είναι το δωμάτιο στο οποίο βρίσκονται οι πίνακες του μουσείου. Είναι ένα ψηλό δωμάτιο με έναν ξύλινο ψευδο-όροφο ο οποίος συνδέεται με το πάτωμα μέσω μιας ξύλινης σκάλας. Σε αμφότερα τα επίπεδα υπάρχουν αλληλεπιδραστικοί πίνακες αλλά και διακοσμητικά λάβαρα. Στον 2ο όροφο υπάρχει μια πόρτα για το room_magic ενώ στον κάτω όροφο μια πόρτα για το room_church.
- **Room_Church:** είναι ένα απλό δωμάτιο με 2 σειρές κολώνες και αψιδωτό θόλο.
- **Room_Magic:** αυτό το εξαγωνικό δωμάτιο εκτείνεται σε μεγάλη απόσταση τόσο προς τα πάνω όσο και προς τα κάτω. Στο επίπεδο της πόρτας που ενώνει το δωμάτιο αυτό με το room_gallery υπάρχει μια μεγάλη πέτρινη ράμπα με μια τρύπα μεγάλης διαμέτρου στο κέντρο της. Υπάρχει ένα πάνελ από πέτρα σε αυτή τη ράμπα το οποίο έχει έξι κουμπιά με τα οποία μπορεί ο χρήστης να δημιουργήσει εφέ με φώτα. Πάτημα ενός κουμπιού θα δημιουργήσει ένα φως που θα ταξιδέψει από την κορυφή του δωματίου ως το κέντρο του, δηλαδή στο επίπεδο της ράμπας και θα υλοποιήσει κάποιο εφέ.




**ΕΙΚΟΝΑ 6:** ΚΑΤΟΨΗ ΤΟΥ ΧΑΡΤΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ



**Εικονα 7:** Τρισδιαστατη οψη του Χαρτη της Εφαρμογης

# ΚΕΦΑΛΑΙΟ 6:
# Αντικείμενα αλληλεπίδρασης

## 6.1: Κλειδί (Key.cs)

```
datablock StaticShapeData(DB_key){
  className = "Key";
  category = "Key";
  shapefile = "~/data/active/key/key.dts";
};

function Key::onAdd(%this,%obj){}
function Key::onCollision(%this,%obj,%col,%vec,%speed){}
function Key::onRemove(%this, %obj){}

function Key::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has picked up key " @ %obj.getID() );
  commandToClient( %client, 'writeHint', "You have picked up the " @ %obj.name @ ".");
  if( %client.inventory ){
    %client.inventory.setScale( %client.inventoryXscale );
    %client.inventory = 0;
  }
  %client.inventory = %obj.getID();
  %client.inventoryXscale = %obj.getScale();
  %obj.setScale("0 0 0");
  commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/invkey.png");
}
```

Η κλάση key έχει ουσιαστικά μόνο μια συνάρτηση, την activate. Η activate καλείται όταν ο χρήστης ενεργοποιήσει (με πάτημα του αριστερού πλήκτρου του ποντικιού) ένα αντικείμενο της κλάσης key. Στην εφαρμογή τα κλειδιά έχουν μια παραπάνω μεταβλητή (εκτός από αυτές που καθορίζουν τη θέση, το DataBlock, το όνομα κτλ.), την unlocks: Το όρισμα της unlocks είναι το ID της πόρτας που ξεκλειδώνει από αυτό το κλειδί (π.χ. unlocks = “door5”). Η activate ξεκινά ενημερώνοντας τον χρήστη (μέσω του γραφικού περιβάλλοντος του πελάτη) ότι πήρε το κλειδί. Αν ο χρήστης έχει ήδη ένα αντικείμενο στο inventory του τότε η εφαρμογή επαναφέρει το αντικείμενο αυτό στην αρχική του θέση. Κατόπιν η μεταβλητή inventory του πελάτη παίρνει την τιμή του ID του κλειδιού, σώζει την παρούσα θέση του κλειδιού στον κόσμο και το αντικείμενο του κλειδιού «εξαφανίζεται» από τον ψηφιακό κόσμο, αφού η κλίμακά του μηδενίζεται. Τέλος, το στοιχείο GunSight του γραφικού περιβάλλοντος του πελάτη αλλάζει σε ένα διαφανές PNG που απεικονίζει ένα κλειδί (έτσι ώστε ο χρήστης να ξέρει ότι στο inventory του έχει ένα κλειδί.

**Η ΈΝΝΟΙΑ ΤΟΥ INVENTORY**

Κάθε χρήστης μπορεί να πάρει ένα αντικείμενο για να το χρησιμοποιήσει σε κάποιο άλλο σημείο της εφαρμογής (στο παραπάνω παράδειγμα, το προφανές ζευγάρι είναι το κλειδί και η πόρτα). Προφανώς τα αντικείμενα που μπορεί να πάρει ο χρήστης είναι συγκεκριμένα, και τα σημεία στα οποία μπορούν να χρησιμοποιηθούν ακόμα πιο συγκεκριμένα. Για να μπορεί ο χρήστης να μεταφέρει ένα αντικείμενο, προστίθεται η έννοια του inventory στην εφαρμογή. Η μεταβλητή inventory είναι στοιχείο του πελάτη και συνοδεύεται από την βοηθητική μεταβλητή inventoryXscale. Η διαδικασία που ακολουθείται για να πάρει ο χρήστης ένα αντικείμενο είναι η εξής: η inventory παίρνει την τιμή του ID του αντικειμένου που χρησιμοποιεί ο χρήστης. Ταυτόχρονα η inventoryXscale παίρνει την τιμή του scale του αντικειμένου. Κατόπιν το αντικείμενο που παίρνει ο χρήστης «εξαφανίζεται» από την εφαρμογή, αλλά δεν παύει να υπάρχει σε αυτήν. Αυτό επιτυγχάνεται αν μηδενίσουμε το μέγεθος του αντικειμένου: το αντικείμενο είναι ακόμα

εκεί, απλά είναι πολύ μικρό και δε φαίνεται. Επίσης αλλάζει το GUI του πελάτη για να είναι εμφανές ότι ο χρήστης έχει κάτι στο inventory του, και τι είναι αυτό. Αυτό επιτυγχάνεται αλλάζοντας την εικόνα του στοιχείου Gunsight σε κάτι που απεικονίζει το αντικείμενο στο inventory.
Για να ελευθερώσει ο χρήστης το inventory του, είτε επειδή παίρνει ένα νέο αντικείμενο είτε πατώντας το κουμπί drop, το αντικείμενο που υπάρχει στο inventory επαναφέρεται στο αρχικό του μέγεθος, και το inventory μηδενίζεται. Ταυτόχρονα το Gunsight αλλάζει και πάλι για να δείξει ότι ο χρήστης δεν έχει πια το αντικείμενο αυτό στο inventory.
Μια ειδική περίπτωση είναι όταν το αντικείμενο χρησιμοποιηθεί και δεν χρειάζεται πια (για παράδειγμα, αν ξεκλειδώσει μια πόρτα τότε το κλειδί δε χρειάζεται πια). Σ' αυτήν την περίπτωση αντί να επαναφέρουμε το αντικείμενο του inventory στο αρχικό του μέγεθος, το απομακρύνουμε πλήρως από την εφαρμογή με την εντολή delete.

## 6.2: Πόρτες (Door.cs)

```
datablock StaticShapeData(DB_door){
  className = "Swinging_Door";
  category = "Door";
  shapefile = "~/data/active/door/door.dts";
};

datablock StaticShapeData(DB_doubledoor){
  className = "Exit_Door";
  category = "Door";
  shapefile = "~/data/active/door/double_door.dts";
};

// GENERIC DOOR FUNCTIONS

function Door::onAdd(%this,%obj){}
function Door::onCollision(%this,%obj,%col,%vec,%speed){}
function Door::onRemove(%this, %obj){}

// SWINGING DOOR FUNCTIONS

function Swinging_Door::Activate(%this, %obj, %client){
  if(%obj.getDataBlock().checkStuck(%obj,%client)){
    return;
  }
  if(%obj.state $= "closed"){               // if it is closed, open it...
    Echo("Opening door " @ %obj.getID() @ "!");
    %obj.playThread(0,"open");
    %obj.state = "opening";
    %this.schedule(2000, "changeState", %obj);
  }
  if(%obj.state $= "open"){                 // if it is open, close it...
    Echo("Closing door " @ %obj.getID() @ "!");
    %obj.playThread(0,"close");
    %obj.state = "closing";
    %this.schedule(2000, "changeState", %obj);
  }
  if(%obj.state $= "locked"){               // if it is locked, it can't be opened.
    if( ( %client.inventory ) && ( %client.inventory.getDatablock().category $= "Key" )
&& ( %client.inventory.unlocks $= %obj.getName() ) ){
      Echo("Door " @ %obj.getID() @ " has been unlocked.");
      commandToClient( %client, 'writeHint', "This door is unlocked.");
      %client.inventory = 0;
      commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
      %obj.playThread(0,"open");
      %obj.state = "opening";
      %this.schedule(2000, "changeState", %obj);
    } else {
      Echo("Door " @ %obj.getID() @ " is locked. It can't be opened!");
      if( ( %client.inventory ) && ( %client.inventory.getDatablock().category $= "key" )
){
       commandToClient( %client, 'writeHint', "The key does not fit this door.");
        // %client.inventory.setScale( %client.inventoryXscale );
        // %client.inventory = 0;
        // commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
      } else {
       commandToClient( %client, 'writeHint', "This door is locked.");
      }
    }
  }
}

function Swinging_Door::changeState(%this, %obj){
  Error("Door Usable!");
  if( %obj.state $= "opening"){
    %obj.state = "open";
  }
```

```
  if( %obj.state $= "closing"){
    %obj.state = "closed";
  }
}

function Swinging_Door::checkStuck(%this, %obj, %client){
  %player_xform = %client.player.getTransform();
  %player_x = GetWords(%player_xform,0);
  %player_y = GetWords(%player_xform,1);
  %player_z = GetWords(%player_xform,2);
  %obj_xform = %obj.getTransform();
  %obj_x = GetWords(%obj_xform,0);
  %obj_y = GetWords(%obj_xform,1);
  %obj_z = GetWords(%obj_xform,2);
  %d_x = %obj_x - %player_x;
  %d_y = %obj_y - %player_y;
  %d_z = %obj_z - %player_z;
  //Echo("(dx dy dz) = (" @ %d_x SPC %d_y SPC %d_z @ ")");
  if( ( (%d_x > -1.26 ) & (%d_x < 1 ) ) &            // defining a "stuck" boundary box
      ( (%d_y > -1.4 ) & (%d_y < 1.2 ) ) &
      ( (%d_z > 0 ) & (%d_z < 3 ) ) ){
    Error("Door Stuck! Get out of the way...");
    commandToClient( %client, 'writeHint', "Door Stuck! Get out of its way...");
    return(1);
  }
  return(0);
}

// EXIT DOOR FUNCTIONS

function Exit_Door::Activate(%this, %obj, %client){
  if(%obj.state $= "closed"){
    MessageBoxYesNo( "Quit Window", "Exit Museum Walkthrough?", "quit();","");
  }
  if(%obj.state $= "locked"){
    if( ( %client.inventory ) && ( %client.inventory.getDatablock().category $= "Key" )
&& ( %client.inventory.unlocks $= %obj.getName() ) ){
      Echo("Door " @ %obj.getID() @ " has been unlocked.");
      commandToClient( %client, 'writeHint', "This door is unlocked.");
      %client.inventory = 0;
      commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
      %obj.state = "closed";
    } else {
      Echo("Door " @ %obj.getID() @ " is locked. It can't be opened!");
      if( ( %client.inventory ) && ( %client.inventory.getDatablock().category $= "key" )
){
       commandToClient( %client, 'writeHint', "The key does not fit this door.");
      } else {
       commandToClient( %client, 'writeHint', "This door is locked.");
      }
    }
  }
}
```

Στην εφαρμογή υπάρχουν δύο είδη πόρτας: η Swinging_Door που είναι ένα animated μοντέλο που ανοιγοκλείνει, και η Exit_Door που κλείνει την εφαρμογή αν χρησιμοποιηθεί.

Κάθε Swinging_Door έχει μια παραπάνω μεταβλητή (εκτός από αυτές που καθορίζουν τη θέση, το DataBlock, το όνομα κτλ.), την state: Το όρισμα της state μπορεί να είναι “locked”, “open”, “opening”, “closed”, “closing”. Όταν χρησιμοποιείται μια πόρτα Swinging_Door καλείται η activate: αρχικά ελέγχει αν υπάρχει χώρος να ανοίξει η πόρτα, καλώντας την check_Stuck. Η check_Stuck επιστρέφει 1 (και ενημερώνει τον πελάτη μέσω του WriteHint) σε περίπτωση που ο avatar του χρήστη βρίσκεται μέσα στα όρια ενός κύβου από την πλευρά που ανοίγει η πόρτα. Αν η check_Stuck επιστρέψει 1, η activate τερματίζεται. Αν όχι, ελέγχεται η μεταβλητή state της πόρτας: αν είναι “closed” τότε ξεκινάει το animation ανοίγματος (με χρήση της playThread(0,”open”) ) και το state

αλλάζει σε ”opening”. Το animation ανοίγματος διαρκεί λίγο λιγότερο από 20sec, γι’ αυτό και με το πέρας 20sec (με την εντολή Schedule) καλείται η εντολή changeState που αλλάζει το state της πόρτας από “opening” σε “open” (επιτρέποντας έτσι το χρήστη να την κάνει activate για να την κλείσει). Η ίδια διαδικασία ακολουθείται αν το state είναι “open”: ξεκινάει το animation κλεισίματος, το state γίνεται “closing” και μετά από 20sec αλλάζει και πάλι σε “closed”. Στην activate δεν υπάρχουν έλεγχοι για τα states “opening” και “closing” γιατί πρέπει να κλείσει η πόρτα πριν μπορέσει ο χρήστης να την ανοίξει (και αντιστρόφως). Ο τελικός έλεγχος είναι για το state “locked”: η πόρτα που είναι κλειδωμένη ανοίγει μόνο αν ο χρήστης έχει στο inventory του ένα κλειδί που έχει ως μεταβλητή “unlocks” το ID της πόρτας αυτής. Σ’ αυτήν την περίπτωση το κλειδί απομακρύνεται από το inventory και από την εφαρμογή (με τη συνάρτηση delete) και ακολουθείται η ίδια διαδικασία με την περίπτωση του state “closed”. Αν ο χρήστης δεν έχει κλειδί στο inventory ή η μεταβλητή unlocks του κλειδιού δεν συμπίπτει με το ID της πόρτας, ενημερώνεται ο χρήστης με εντολή WriteHint προς τον πελάτη και η activate τερματίζεται.

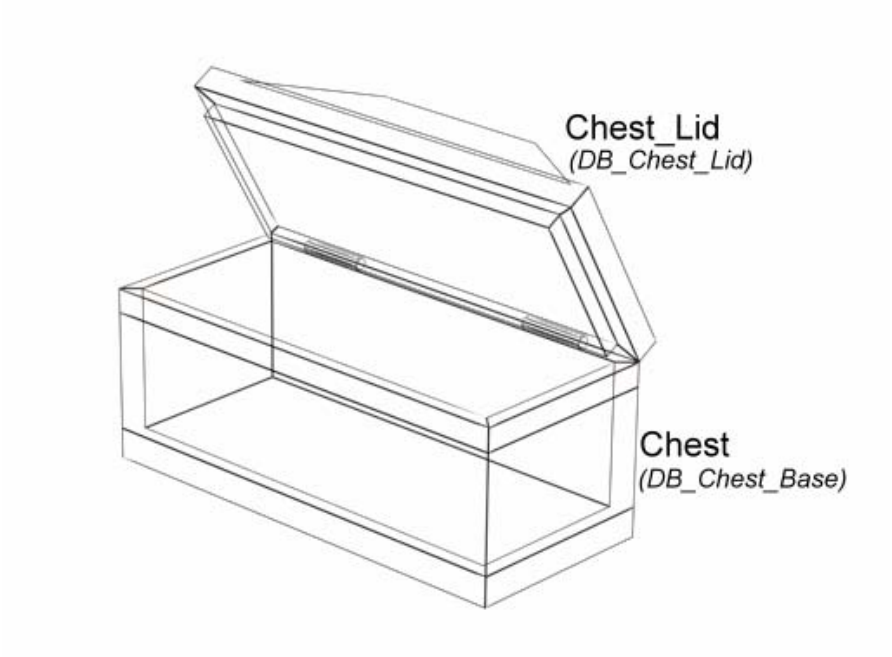
Η Exit_Door χρησιμοποιεί μόνο τη συνάρτηση activate η οποία ελέγχει το state της πόρτας (όπως και η Swinging_Door). Αν είναι “closed” τότε ο χρήστης ερωτάται αν θέλει να βγει από την εφαρμογή με ένα παράθυρο τύπου Yes/No. Αν επιλέξει Yes η εφαρμογή τερματίζεται. Αν η πόρτα Exit_Door έχει state “locked” τότε ακολουθείται η ίδια διαδικασία με την Swinging_Door και αν ο χρήστης έχει κλειδί με μεταβλητή unlocks που ταιριάζει στην πόρτα, τότε το state αλλάζει σε “closed”. Η Exit_Door δεν έχει animation και έτσι δεν κάνει χρήση των states “closing”, “opening” και “open”.

## 6.3: Containers (Chest.cs και Desk.cs)

```
datablock StaticShapeData(DB_chest_base){
  className = "Chest";
  category = "Chest";
  shapefile = "~/data/active/chest/chest_base.dts";
};

datablock StaticShapeData(DB_chest_lid){
  className = "Chest_Lid";
  category = "Chest";
  shapefile = "~/data/active/chest/chest_lid.dts";
};

function Chest_Lid::onAdd(%this,%obj){}
function Chest_Lid::onCollision(%this,%obj,%col,%vec,%speed){}
function Chest_Lid::onRemove(%this, %obj){}

function Chest_Lid::Activate(%this, %obj, %client){
  if(%obj.state $= "closed"){                        // if it is closed, open it...
    Echo("Opening chest " @ %obj.getID() @ "!");
    %obj.playThread(0,"open");
    %obj.state = "opening";
    %this.schedule(1000, "changeState", %obj);
    if(%obj.contains $= ""){
      commandToClient( %client, 'writeHint', "The chest is empty." );
    } else {
      %new_obj = new StaticShape( GetWord(%obj.contains_name, 0 ) @ "_1" ) {
        dataBlock = %obj.contains;
        name = %obj.contains_name;
      };
      %new_obj.setScale( %obj.contains_scale );
      %new_obj.setTransform( %obj.contains_Xform );
      %new_obj.getDataBlock().activate(%new_obj, %client);
      // now the container is empty
      %obj.contains = "";
    }
  }
  if(%obj.state $= "open"){                          // if it is open, close it...
    Echo("Closing chest " @ %obj.getID() @ "!");
    %obj.playThread(0,"close");
    %obj.state = "closing";
    %this.schedule(1000, "changeState", %obj);
  }
  if(%obj.state $= "locked"){                        // if its locked, it can't be opened.
    if( ( %client.inventory ) && ( %client.inventory.getDatablock().category $= "Key" )
&& ( %client.inventory.unlocks $= %obj.getName() ) ){
      Echo("Chest " @ %obj.getID() @ " has been unlocked.");
      commandToClient( %client, 'writeHint', "The chest is unlocked." );
      commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png" );
      %client.inventory = 0;
      %obj.playThread(0,"open");
      %obj.state = "opening";
      %this.schedule(1000, "changeState", %obj);
      if(%obj.contains $= ""){
        commandToClient( %client, 'writeHint', "The chest is empty." );
      } else {
        %new_obj = new StaticShape( GetWord(%obj.contains_name, 0 ) SPC 1 ) {
          dataBlock = %obj.contains;
          name = %obj.contains_name;
        };
        %new_obj.setScale( %obj.contains_scale );
        %new_obj.setTransform( %obj.contains_Xform );
        %new_obj.getDataBlock().activate(%new_obj, %client);
        // now the container is empty
        %obj.contains = "";
      }
    } else {
      Echo("Chest " @ %obj.getID() @ " is locked. It can't be opened!");
```

```
      commandToClient( %client, 'writeHint', "This chest is locked.");
    }
  }
}

function Chest_Lid::changeState(%this, %obj){
  Error("Chest Usable!");
  if( %obj.state $= "opening"){
    %obj.state = "open";
  }
  if( %obj.state $= "closing"){
    %obj.state = "closed";
  }
}
```

Το σεντούκι αποτελείται από 2 μοντέλα (και άρα και 2 StaticShape DataBlocks): τη βάση (DB_chest_base), που δεν είναι animated και το καπάκι (DB_chest_lid) που έχει animation να ανοιγοκλείνει. Οι πληροφορίες (όπως το τι περιέχει και ποιο κλειδί το ανοίγει) αναφέρονται στο καπάκι. Στην συνάρτηση ServerCmdActivate που αναλύσαμε στο **Κεφάλαιο 5: Αλληλεπίδραση με τον Χρήστη** όταν ενεργοποιείται αντικείμενο της κλάσης Chest_Base, τότε απ' ευθείας στέλνεται εντολή να ενεργοποιηθεί το καπάκι του αντικειμένου αυτού. Έτσι αρκεί να ορίσουμε συναρτήσεις για την κλάση Chest_lid: η activate ελέγχει τη μεταβλητή state του chest_lid (η οποία είναι όμοια με την state της κλάσης swinging_door). Αν είναι κλειστό ("closed") παίζει το animation ανοίγματος (που διαρκεί λιγότερο από 10sec), αλλάζει το state σε "opening" και μετά από 10sec αλλάζει σε "open" (με χρήση της schedule με όρισμα changeState) με αποτέλεσμα να γίνεται και πάλι χρησιμοποιήσιμο. Εδώ διαφέρει ο κώδικας από τον αντίστοιχο της πόρτας: όταν ανοίγει το σεντούκι, ελέγχεται μια ακόμα μεταβλητή του chest_lid: η contains η οποία αναφέρεται στο περιεχόμενο του σεντουκιού. Για τις ανάγκες του περιεχομένου υπάρχουν οι μεταβλητές contains, contains_Xform, contains_name, contains_scale με προφανείς ιδιότητες η καθεμία. Όταν η contains δεν είναι κενή, τότε υπάρχει πράγματι αντικείμενο μέσα στο container, και δημιουργείται το αντικείμενο αυτό με χρήση της new και ορίσματα αυτά που σώζονται στα contains_... (δηλαδή το DataBlock σώζεται στο contains, το όνομα του αντικειμένου στο contains_name, τη θέση στην αποστολή στο contains_Xform και το μέγεθος στο contains_scale). Μια τελευταία κίνηση του κώδικα είναι να ενεργοποιήσει το νέο αντικείμενο (με εντολή activate). Επειδή το περιεχόμενο του σεντουκιού είναι αντικείμενο που όταν ενεργοποιηθεί τοποθετείται στο intentory του χρήστη και «εξαφανίζεται» από τον ψηφιακό κόσμο, με τον παραπάνω τρόπο δίνεται η ψευδαίσθηση στον χρήστη ότι ανοίγοντας το container πήρε το αντικείμενο που βρισκόταν μέσα.



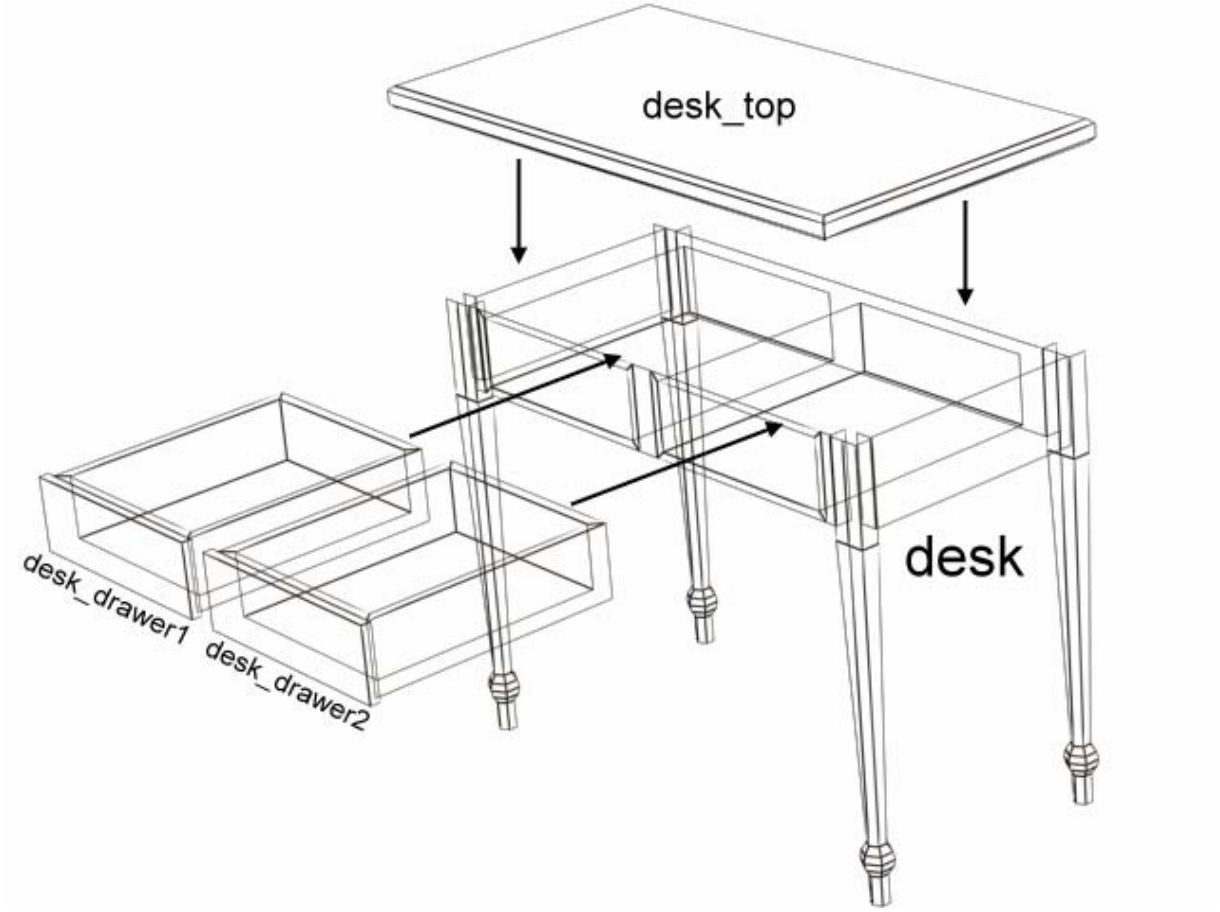

**ΕΙΚΟΝΑ 8:** ΤΑ ΑΝΤΙΚΕΙΜΕΝΑ ΤΟΥ CHEST (ΣΕ ΠΑΡΕΝΘΕΣΗ ΤΑ DATABLOCKS)

Αν το state είναι ανοιχτό ("open") εφαρμόζεται ίδια διαδικασία με animation κλεισίματος, αλλαγή state σε "closing" και μετά από 10sec αλλαγή state σε "closed" (χωρίς να ασχολείται η εφαρμογή με την contains κτλ.). Αν το αντικείμενο chest_lid είναι κλειδωμένο (state "locked") τότε εφαρμόζεται η ίδια διαδικασία που αναλύθηκε στις πόρτες της κλάσης swinging_Door. Το σεντούκι που είναι κλειδωμένο ανοίγει μόνο αν ο χρήστης έχει στο inventory του ένα κλειδί που έχει ως μεταβλητή "unlocks" το ID του chest_lid του σεντουκιού αυτού. Σ' αυτήν την περίπτωση το κλειδί απομακρύνεται από το inventory και από την εφαρμογή (με τη συνάρτηση delete) και ακολουθείται η ίδια διαδικασία με την περίπτωση του state "closed". Αν ο χρήστης δεν έχει κλειδί στο inventory ή η μεταβλητή unlocks του κλειδιού δεν συμπίπτει με το ID του σεντουκιού, ενημερώνεται ο χρήστης με εντολή WriteHint προς τον πελάτη και η activate τερματίζεται.

```
// +-------------------------------------------+
// | DESK DRAWERS OPEN/CLOSE                   |
// +-------------------------------------------+

datablock StaticShapeData(DB_desk_top){
  className = "Desk_Top";
  category = "";
  shapefile = "~/data/active/desk/desk_top.dts";
};

datablock StaticShapeData(DB_desk_drawer1){
  className = "Desk_Drawer";
  category = "Desk";
  shapefile = "~/data/active/desk/drawer1.dts";
};

datablock StaticShapeData(DB_desk_drawer2){
  className = "Desk_Drawer";
  category = "Desk";
  shapefile = "~/data/active/desk/drawer2.dts";
};

function Desk_Drawer::onAdd(%this,%obj){}
function Desk_Drawer::onCollision(%this,%obj,%col,%vec,%speed){}
function Desk_Drawer::onRemove(%this, %obj){}
```

```
function Desk_Drawer::Activate(%this, %obj, %client){
  if(%obj.state $= "closed"){                          // if it is closed, open it...
    Echo("Opening Desk Drawer " @ %obj.getID() @ "!");
    %obj.playThread(0,"open");
    %obj.state = "opening";
    %this.schedule(800, "changeState", %obj);
    if(%obj.contains $= ""){
      commandToClient( %client, 'writeHint', "The drawer is empty.");
    } else {
      %new_obj = new StaticShape( GetWord(%obj.contains_name, 0 ) SPC 1 ) {
        dataBlock = %obj.contains;
        name = %obj.contains_name;
      };
      %new_obj.setScale( %obj.contains_scale );
      %new_obj.setTransform( %obj.contains_Xform );
      %new_obj.getDataBlock().activate(%new_obj, %client);
      // now the container is empty
      %obj.contains = "";
    }
  }
  if(%obj.state $= "open"){                            // if it is open, close it...
    Echo("Closing Desk Drawer " @ %obj.getID() @ "!");
    %obj.playThread(0,"close");
    %obj.state = "closing";
    %this.schedule(800, "changeState", %obj);
  }
  if(%obj.state $= "locked"){                          / if its locked, it can't be opened.
    if( ( %client.inventory ) && ( %client.inventory.getDatablock().category $= "Key" )
&& ( %client.inventory.unlocks $= %obj.getName() ) ){
      Echo("Desk Drawer " @ %obj.getID() @ " has been unlocked.");
      commandToClient( %client, 'writeHint', "This desk drawer is unlocked.");
      %client.inventory = 0;
      commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
      %obj.playThread(0,"open");
      %obj.state = "opening";
      %this.schedule(800, "changeState", %obj);
      if(%obj.contains $= ""){
        commandToClient( %client, 'writeHint', "The drawer is empty.");
      } else {
        %new_obj = new StaticShape( GetWord(%obj.contains_name, 0 ) @ "_1" ) {
          dataBlock = %obj.contains;
          name = %obj.contains_name;
        };
        %new_obj.setScale( %obj.contains_scale );
        %new_obj.setTransform( %obj.contains_Xform );
        %new_obj.getDataBlock().activate(%new_obj, %client);
        // now the container is empty
        %obj.contains = "";
      }
    } else {
      Echo("Desk Drawer " @ %obj.getID() @ " is locked. It can't be opened!");
      commandToClient( %client, 'writeHint', "This desk drawer is locked.");
    }
  }
}

function Desk_Drawer::changeState(%this, %obj){
  Error("Desk Drawer Usable!");
  if( %obj.state $= "opening"){
    %obj.state = "open";
  }
  if( %obj.state $= "closing"){
    %obj.state = "closed";
  }
}
```

Ο παραπάνω κώδικας βρίσκεται στο τέλος του αρχείου desk.cs που εκτός από το γραφείο και τα συρτάρια του (που είναι containers παρόμοια με το σεντούκι) περιέχει και όλα τα είδη γραφείου που θα περιγραφούν στο σκέλος 6.6. Το γραφείο αποτελείται από τέσσερα μέρη, ένα στατικό αντικείμενο TSStatic που δεν είναι αλληλεπιδραστικό και αποτελεί τα πόδια και την επένδυση του γραφείου, ένα αλληλεπιδραστικό αντικείμενο Desk_Top που αντιπροσωπεύει την επιφάνεια εργασίας και 2 συρτάρια που είναι animated και αλληλεπιδραστικά. Μόνο τα συρτάρια, που είναι και τα δύο της κλάσης Desk_Drawer έχουν συναρτήσεις activate και changeState οι οποίες είναι όμοιες με αυτές του Chest_Lid που αναλύθηκε παραπάνω, με ελάχιστες διαφορές (όπως ότι ο χρόνος για να αλλάξει το state από “opening” σε “open” είναι 8sec).




**ΕΙΚΟΝΑ 9:** ΤΑ ΑΝΤΙΚΕΙΜΕΝΑ ΠΟΥ ΑΠΑΡΤΙΖΟΥΝ ΤΟ ΓΡΑΦΕΙΟ

## 6.4: Λάμπες (Lamp.cs)

```
datablock StaticShapeData(DB_lamp1){
  className = "Overhead_Lamp";
  category = "Lamp";
  shapefile = "~/data/active/lamp/lamp.dts";
};

datablock StaticShapeData(DB_torch){
  className = "Torch";
  category = "Lamp";
  shapefile = "~/data/active/lamp/torch.dts";
};


datablock StaticShapeData(DB_floor_lamp){
  className = "Floor_Lamp";
  category = "Lamp";
  shapefile = "~/data/active/lamp/floor_lamp.dts";
};

function Lamp::onAdd(%this,%obj){}
function Lamp::onCollision(%this,%obj,%col,%vec,%speed){}
function Lamp::onRemove(%this, %obj){}
function Lamp::activate(%this,%obj, %client){}

function Torch::onAdd(%this,%obj){
  if(%obj.state $= "lit"){
    %obj.getDatablock().light(%obj);
  }
}

function Torch::activate(%this, %obj, %client){
  if(%obj.state $= "lit"){
    Echo("Putting out torch " @ %obj.getID());
    %obj.state = "unlit";
    %fire = %obj.getName() @ "_fire";
    %fire.delete();
  } else {
    Echo("Lighting torch " @ %obj.getID());
    %obj.state = "lit";
    %obj.getDatablock().light(%obj);
  }
}

function Torch::light(%this, %obj){
  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
    scale = "1 1 1";
    dataBlock = "FireParticleEmitterNode";
    emitter = "FireParticleEmitter";
    velocity = "1";
  };
  %obj_xform = %obj.getTransform();
  %obj_x = GetWord(%obj_xform,0);
  %obj_y = GetWord(%obj_xform,1);
  %obj_z = GetWord(%obj_xform,2);
  %dirA = GetWord(%obj_xform,5);                                    // direction
  %rotA = GetWord(%obj_xform,6);                                    // angle
  %sinA = mSin(%rotA);
  %cosA = mCos(%rotA);

  if( %dirA > 0 ){
    %sinA = -%sinA;
  }

  %end_x = %obj_x + (0.35 * %cosA);
  %end_y = %obj_y + (0.35 * %sinA);
  %end_z = %obj_z + 0.1;
  %vector = %end_x SPC %end_y SPC %end_z;
  %fire.setTransform(%vector SPC "1 0 0 0");
```

```
}

function Floor_Lamp::onAdd(%this, %obj){
  if(%obj.state $= "lit"){
    %obj.getDatablock().light(%obj);
  }
}

function Floor_Lamp::activate(%this, %obj, %client){
  if(%obj.state $= "lit"){
    Echo("Putting out torch " @ %obj.getID());
    %obj.state = "unlit";
    %fire = %obj.getName() @ "_fire";
    %fire.delete();
  } else {
    Echo("Lighting torch " @ %obj.getID());
    %obj.state = "lit";
    %obj.getDatablock().light(%obj);
  }
}

function Floor_Lamp::light(%this, %obj){
  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
    scale = "1 1 1";
    dataBlock = "FireParticleEmitterNode";
    emitter = "FireParticleEmitter";
    velocity = "1";
  };
  %obj_xform = %obj.getTransform();
  %obj_x = GetWord(%obj_xform,0);
  %obj_y = GetWord(%obj_xform,1);
  %obj_z = GetWord(%obj_xform,2);

  %fire_z = %obj_z + 2.5;
  %vector = %obj_x SPC %obj_y SPC %fire_z;
  %fire.setTransform(%vector SPC "1 0 0 0");
}
```

Οι λάμπες της εφαρμογής είναι αλληλεπιδραστικά αντικείμενα που όταν χρησιμοποιούνται παράγουν ή σβήνουν ένα ParticleEmitter που εμφανίζεται σαν φωτιά. Κάθε αντικείμενο που είναι της κατηγορίας Lamp έχει μια μεταβλητή state η οποία μπορεί να πάρει τις τιμές “lit” και “unlit”. Όταν ξεκινά η εφαρμογή ελέγχεται το state όταν προστίθεται το αντικείμενο στην αποστολή (δηλαδή στην OnAdd) και αν είναι “lit” δημιουργείται ένα ParticleEmitter με ID αυτό της λάμπας με την κατάληξη “_fire” (π.χ. αν η λάμπα έχει ID Torch1 τότε το ParticleEmitter θα έχει ID Torch1_fire) και τοποθετείται στην κατάλληλη θέση ως προς την λάμπα (για τους πυρσούς –κλάση Torch– η θέση εξαρτάται από τη γωνιά που είναι τοποθετημένος ο πυρσός, ενώ για το Floor_Lamp η θέση είναι σταθερή και ανεξάρτητη από τη γωνία). Όταν ένα αντικείμενο της κατηγορίας Lamp (είτε είναι Torch είτε Floor_Lamp) καλεί την activate, ελέγχεται το state της λάμπας, και αν είναι “unlit” δημιουργείται ο ParticleEmitter, τοποθετείται στην κατάλληλη θέση (όπως στην onAdd) και το state αλλάζει σε “lit”. Αν η λάμπα είναι “lit” τότε βρίσκουμε τον ParticleEmitter που της αντιστοιχεί (αναζητούμε το ID της λάμπας με το επίθεμα “_fire”) και το διαγράφουμε από την εφαρμογή με την εντολή delete.

**PARTICLES ΚΑΙ PARTICLEEMITTERS**

Υπάρχουν δύο βασικοί τρόποι να υλοποιηθούν particle effects στο Torque. Ο πρώτος τρόπος είναι ένας αυτόνομος κόμβος (αποκαλούμενος ParticleEmitterNode) που εκπέμπει τα Particles από μια ορισμένη θέση. Ο δεύτερος τρόπος είναι ο εκπομπός (αποκαλούμενος ParticleEmitter) που συνδυάζεται λειτουργικά με ένα αντικείμενο. Σ' αυτήν την εφαρμογή χρησιμοποιείται εκτεταμένα ο πρώτος τρόπος. Η διαδικασία περιλαμβάνει τη δημιουργία ενός νέου αρχείου για τα Particles (με όνομα Fire.cs) που περιέχει ParticleEmitterData, ParticleData, ParticleEmitterNodeData, και για καθένα από τα είδη «φωτιάς» που θέλουμε να χρησιμοποιήσουμε στην εφαρμογή. Τα παραπάνω Data συνδέονται σε ένα νέο αντικείμενο, ParticleEmitterNode. Στο αρχείο Lamp.cs κατασκευάζονται ParticleEmitterNodes που χρησιμοποιούν τους FireParticleEmitterNode και FireParticleEmitter. Από την άλλη η φωτιά του κεριού (από τα είδη γραφείου) ορίζεται μέσα στο αρχείο αποστολής museum_map.mis με CandleParticleEmitterNode και CandleParticleEmitter.
Τα ParticleEmitterNodes είναι χρήσιμα για ποικίλα αποτελέσματα από τα φώτα Νέον ως τις πυρκαγιές και τα ολογράμματα. Παρόλο που οι κόμβοι είναι στάσιμοι, φαίνονται «ζωντανοί» καθώς η κίνηση των particles που παράγουν είναι τυχαία και ως εκ τούτου η μορφή του ParticleEmitterNode αλλάζει συνέχεια. Τα Nodes αυτά τοποθετούνται στον κόσμο με την προσθήκη ενός DataBlock στο αρχείο αποστολής ή δυναμικά με χρήση της new σε άλλα τμήματα του κώδικα (όπως το Lamp.cs). Οι παράμετροι στα ParticleEmitterNodeData, ParticleEmitterData, και ParticleData είναι πολλές και διάφορες, και με μικρές αλλαγές σε αυτές μπορούμε να δημιουργήσουμε πολλά και ενδιαφέρονται αποτελέσματα.

## 6.5: Βιβλία και πίνακες (Book.cs και Painting.cs)

```
datablock StaticShapeData(DB_book1){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book1.dts";
};

datablock StaticShapeData(DB_book1_hor){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book1_hor.dts";
};

datablock StaticShapeData(DB_book2){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book2.dts";
};

datablock StaticShapeData(DB_book2_hor){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book2_hor.dts";
};

datablock StaticShapeData(DB_book3){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book3.dts";
};

datablock StaticShapeData(DB_book3_hor){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book3_hor.dts";
};

datablock StaticShapeData(DB_book4){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book4.dts";
};

datablock StaticShapeData(DB_book4_hor){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book4_hor.dts";
};

datablock StaticShapeData(DB_book5){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book5.dts";
};

datablock StaticShapeData(DB_book5_hor){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book5_hor.dts";
};

datablock StaticShapeData(DB_book6){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book6.dts";
};

datablock StaticShapeData(DB_book6_hor){
  className = "Book";
```

```
  category = "Book";
  shapefile = "~/data/items/books/book6_hor.dts";
};

datablock StaticShapeData(DB_book7){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book7.dts";
};

datablock StaticShapeData(DB_book7_hor){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book7_hor.dts";
};

datablock StaticShapeData(DB_book8){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book8.dts";
};

datablock StaticShapeData(DB_book8_hor){
  className = "Book";
  category = "Book";
  shapefile = "~/data/items/books/book8_hor.dts";
};

function Book::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " is reading book " @ %obj.getID() );
  %filename = "control/data/items/books/" @ %obj.page;
  commandToClient( %client, 'setGUIbitmap', "PaintingCover", %filename );
  %obj.state = "read";
  if(%obj.getName() $= "Book_FormFilling"){
    commandToClient( %client, 'writeHint', "You acquire the necessary information about
forms" );
  }
  %conn = %client.getControlObject();
  if (%conn == %client.player){
    %transform = %obj.getTransform();
    %conn = %client.camera;
    %conn.mode = "Observe";
    %client.setControlObject(%conn);
  }
}
```

Στην εφαρμογή υπάρχουν βιβλία διαφόρων σχημάτων και χρωμάτων αλλά όλα ανήκουν στην ίδια κλάση και έχουν κοινή συνάρτηση activate. Όταν ενεργοποιηθεί λοιπόν ένα βιβλίο, προστίθεται η μεταβλητή state με όρισμα “read”, ενώ αν είναι το βιβλίο Book_FormFilling που έχει μια ιδιαίτερη σημασία στην εφαρμογή (θα εξηγηθεί στο τμήμα για τα **Είδη Γραφείου**) στέλνεται μήνυμα στον χρήστη με την writeHint. Κατόπιν το αρχείο που περιέχεται στην μεταβλητή page του κάθε βιβλίου τοποθετείται στο στοιχείο PaintingCover του GUI του πελάτη και ο χρήστης χάνει τον έλεγχο του avatar έτσι ώστε να μην μπορεί να τον μετακινεί ενώ διαβάζει το κείμενο που εμφανίζεται στην οθόνη (θυμίζουμε ότι το PaintingCover καλύπτει όλη την οθόνη). Ο μόνος τρόπος να εξαφανιστεί το PaintingCover και να επανέρθει ο έλεγχος του avatar στον χρήστη είναι με πάτημα του πλήκτρου Q για να ενεργοποιηθεί η εντολή “QuitOrbit” στον εξυπηρετητή (όπως αναλύσαμε στο **Κεφάλαιο 5: Αλληλεπίδραση με τον χρήστη**).

```
datablock StaticShapeData(DB_painting1){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting1/picture.dts";
  image = "control/data/items/paintings/Painting1/background";
};

// NEW ADDITIONS FOR GALLERY

datablock StaticShapeData(DB_painting3){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting3/picture.dts";
  image = "control/data/items/paintings/Painting3/background";
};

datablock StaticShapeData(DB_painting4){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting4/picture.dts";
  image = "control/data/items/paintings/Painting4/background";
};

datablock StaticShapeData(DB_painting5){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting5/picture.dts";
  image = "control/data/items/paintings/Painting5/background";
};

datablock StaticShapeData(DB_painting6){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting6/picture.dts";
  image = "control/data/items/paintings/Painting6/background";
};

datablock StaticShapeData(DB_painting7){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting7/picture.dts";
  image = "control/data/items/paintings/Painting7/background";
};

datablock StaticShapeData(DB_painting8){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting8/picture.dts";
  image = "control/data/items/paintings/Painting8/background";
};

datablock StaticShapeData(DB_painting9){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting9/picture.dts";
  image = "control/data/items/paintings/Painting9/background";
};

datablock StaticShapeData(DB_painting10){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting10/picture.dts";
  image = "control/data/items/paintings/Painting10/background";
};

datablock StaticShapeData(DB_painting11){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting11/picture.dts";
  image = "control/data/items/paintings/Painting11/background";
```

```
};

datablock StaticShapeData(DB_painting12){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting12/picture.dts";
  image = "control/data/items/paintings/Painting12/background";
};

datablock StaticShapeData(DB_painting13){
  category = "Painting";
  className = "Painting";
  shapefile = "~/data/items/paintings/Painting13/picture.dts";
  image = "control/data/items/paintings/Painting13/background";
};

function Painting::onAdd(%this,%obj){}
function Painting::onCollision(%this,%obj,%col,%vec,%speed){}
function Painting::onRemove(%this, %obj){}

function Painting::Activate(%this, %obj, %client){
  Echo("Painting activated");
  commandToClient( %client, 'setGUIbitmap', "PaintingCover", %obj.getDataBlock().image);
  commandToClient( %client, 'setGUIvisible', "PaintingCover", 1 );
  %conn = %client.getControlObject();
  if (%conn == %client.player){
    %transform = %obj.getTransform();
    %conn = %client.camera;
    %conn.mode = "Observe";
    %client.setControlObject(%conn);
  }
}
```

Όπως και για τα βιβλία, υπάρχουν πολλοί πίνακες στην εφαρμογή, καθένας από τους οποίους ανήκει σε ένα ξεχωριστό DataBlock. Κάθε DataBlock που περιγράφει έναν πίνακα περιέχει την μεταβλητή image στην οποία αποθηκεύεται η διεύθυνση του αρχείου εικόνας που θα εμφανίσει ο πίνακας όταν ενεργοποιηθεί. Αυτό είναι μια διαφορά από τον κώδικα των βιβλίων, στα οποία η αντίστοιχη μεταβλητή page ανήκε σε ένα αντικείμενο, ενώ εδώ ανήκει σε ένα DataBlock. Ο κώδικας ενεργοποίησης ενός πίνακα είναι μια απλουστευμένη έκδοση του κώδικα της κλάσης Book: το αρχείο που περιέχεται στην μεταβλητή image του κάθε DataBlock τοποθετείται στο στοιχείο PaintingCover του GUI του πελάτη και ο χρήστης χάνει τον έλεγχο του avatar έτσι ώστε να μην μπορεί να τον μετακινεί ενώ εμφανίζεται η εικόνα στην οθόνη (θυμίζουμε ότι το PaintingCover καλύπτει όλη την οθόνη). Ο μόνος τρόπος να εξαφανιστεί το PaintingCover και να επανέρθει ο έλεγχος του avatar στον χρήστη είναι με πάτημα του πλήκτρου Q για να ενεργοποιηθεί η εντολή “QuitOrbit” στον εξυπηρετητή (όπως αναλύσαμε στο **Κεφάλαιο 5: Αλληλεπίδραση με τον Χρήστη**).

## 6.6: Είδη Γραφείου (Desk.cs)

```
// +-----------------------------------------+
// | DESK ITEMS                              |
// +-----------------------------------------+


datablock StaticShapeData(DB_inkquill){
  className = "Inkquill";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/inkquill.dts";
};

datablock StaticShapeData(DB_inkwell){
  className = "Inkwell";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/inkwell.dts";
};

datablock StaticShapeData(DB_paper){
  className = "Paper";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/paper.dts";
  page = "control/data/active/desk/items/background.jpg";
};

datablock StaticShapeData(DB_paper_written){
  className = "Paper_Written";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/paper_written.dts";
  page = "control/data/active/desk/items/background_written";
};

datablock StaticShapeData(DB_envelope_empty){
  className = "Envelope_Empty";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/envelope.dts";
};

datablock StaticShapeData(DB_envelope){
  className = "Envelope";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/envelope.dts";
};

datablock StaticShapeData(DB_envelope_sealed){
  className = "Envelope_Sealed";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/envelope_sealed.dts";
};

datablock StaticShapeData(DB_signet_ring){
  className = "Signet_Ring";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/ring.dts";
};

datablock StaticShapeData(DB_candle){
  className = "Candle";
  category = "Desk_Item";
  shapefile = "~/data/active/desk/items/candle.dts";
};

function Inkquill::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has picked up inkquill " @ %obj.getID() );
  commandToClient( %client, 'writeHint', "You have picked up the " @ %obj.name @ ".");
  if( %client.inventory ){
    %client.inventory.setScale( %client.inventoryXscale );
    %client.inventory = 0;
  }
```

```
  %client.inventory = %obj.getID();
  %client.inventoryXscale = %obj.getScale();
  %obj.setScale("0 0 0");
  commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/invquill.png" );
}

function Inkwell::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has used inkwell " @ %obj.getID() );
  if( ( %client.inventory ) && ( %client.inventory.getDatablock() == DB_inkquill.getID()
) ){
    commandToClient( %client, 'writeHint', "The quill now has enough ink.");
    %client.inventory.state = "inked";
  }
}

function Paper::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has used paper " @ %obj.getID() );
  if( %client.inventory ){
    if( %client.inventory.getDatablock() == DB_envelope_empty.getID() ){
      commandToClient( %client, 'writeHint', "You have not finished the form.");
      return;
    }
    if( %client.inventory.getDatablock() == DB_signet_ring.getID() ){
      commandToClient( %client, 'writeHint', "You need to put it in an envelope first.");
      return;
    }
    if( %client.inventory.getDatablock() == DB_inkquill.getID() ){
      if( %client.inventory.state $= "uninked" ){
        commandToClient( %client, 'writeHint', "The quill doesn't have enough ink.");
        return;
      } else if( %client.inventory.state $= "inked" ){
        if( book_FormFilling.state $= "read" ){
          commandToClient( %client, 'writeHint', "You write down the necessary
information.");
          %new_obj = new StaticShape(%obj.getName() @ "_written") {
            dataBlock = "DB_paper_written";
            name = "Filled Form";
          };
          %new_obj.setTransform( %obj.getTransform() );
          %new_obj.setScale( %obj.getScale() );
          %obj.delete();
          commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
          %client.inventory.setScale( %client.inventoryXscale );
          %client.inventory = 0;
          return;
        } else {
          commandToClient( %client, 'writeHint', "You have no idea how to write this
form.");
          return;
        }
      }
    }
  }
  commandToClient( %client, 'setGUIbitmap', "PaintingCover", %obj.getDatablock().page );
  commandToClient( %client, 'setGUIvisible', "PaintingCover", 0 );
  %conn = %client.getControlObject();
  if (%conn == %client.player){
    %transform = %obj.getTransform();
    %conn = %client.camera;
    %conn.mode = "Observe";
    %client.setControlObject(%conn);
  }
}

function Paper_Written::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has used paper " @ %obj.getID() );
  if( %client.inventory ){
    if( %client.inventory.getDatablock() == DB_inkquill.getID() ){
      commandToClient( %client, 'writeHint', "You have already written all the necessary
```

```
information.");
      return;
    }
    if( %client.inventory.getDatablock() == DB_signet_ring.getID() ){
      commandToClient( %client, 'writeHint', "You need to put it in an envelope first.");
      return;
    }
    if( %client.inventory.getDatablock() == DB_envelope_empty.getID() ){
      commandToClient( %client, 'writeHint', "You put the written paper in the
envelope.");
      //%client.inventory.delete();
      %client.inventory = 0;
      commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
      %new_obj = new StaticShape("envelope") {
        dataBlock = "DB_envelope";
        name = "Envelope";
      };
      %new_obj.setTransform( %obj.getTransform() );
      %new_obj.setScale( %obj.getScale() );
      %obj.delete();
      return;
    }
  }
  commandToClient( %client, 'setGUIbitmap', "PaintingCover", %obj.getDatablock().page);
  commandToClient( %client, 'setGUIvisible', "PaintingCover", 1 );
  %conn = %client.getControlObject();
  if (%conn == %client.player){
    %transform = %obj.getTransform();
    %conn = %client.camera;
    %conn.mode = "Observe";
    %client.setControlObject(%conn);
  }
}

function Envelope_Empty::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has picked up envelope " @ %obj.getID() );
  commandToClient( %client, 'writeHint', "You have picked up an envelope.");
  if( %client.inventory ){
    %client.inventory.setScale( %client.inventoryXscale );
    %client.inventory = 0;
  }
  %client.inventory = %obj.getID();
  %client.inventoryXscale = %obj.getScale();
  %obj.setScale("0 0 0");
  commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/invenvelope1.png");
}

function Envelope::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has picked up envelope " @ %obj.getID() );
  if( ( %client.inventory ) && ( %client.inventory.getDatablock() ==
DB_signet_ring.getID() ) ){
    if( %client.inventory.state $= "waxed" ){
      commandToClient( %client, 'writeHint', "You seal the envelope.");
      %client.inventory.setScale( %client.inventoryXscale );
      %client.inventory = 0;
      commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
      %new_obj = new StaticShape("envelope_sealed") {
        dataBlock = "DB_envelope_sealed";
        name = "Sealed Envelope";
      };
      %new_obj.setTransform( %obj.getTransform() );
      %new_obj.setScale( %obj.getScale() );
      %obj.delete();
    } else {
      commandToClient( %client, 'writeHint', "You cannot seal the envelope without
wax.");
    }
  } else {
```

```
    commandToClient( %client, 'writeHint', "You have picked up an envelope.");
    if( %client.inventory ){
      %client.inventory.setScale( %client.inventoryXscale );
      %client.inventory = 0;
    }
    %client.inventory = %obj.getID();
    %client.inventoryXscale = %obj.getScale();
    %obj.setScale("0 0 0");
    commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/invenvelope1.png");
  }
}

function Envelope_Sealed::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has picked up envelope " @ %obj.getID() );
  commandToClient( %client, 'writeHint', "You have picked up the sealed envelope.");
  if( %client.inventory ){
    %client.inventory.setScale( %client.inventoryXscale );
    %client.inventory = 0;
  }
  %client.inventory = %obj.getID();
  %client.inventoryXscale = %obj.getScale();
  %obj.setScale("0 0 0");
  commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/invenvelope2.png");
}

function Signet_Ring::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has picked up the ring " @ %obj.getID() );
  commandToClient( %client, 'writeHint', "You have picked up the signet ring.");
  if( %client.inventory ){
    %client.inventory.setScale( %client.inventoryXscale );
    %client.inventory = 0;
  }
  %client.inventory = %obj.getID();
  %client.inventoryXscale = %obj.getScale();
  %obj.setScale("0 0 0");
  commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/invring.png");
}

function Candle::activate(%this, %obj, %client){
  Echo("Client " @ %client @ " has used candle " @ %obj.getID() );
  if( ( %client.inventory ) && ( %client.inventory.getDatablock() ==
DB_signet_ring.getID() ) ){
    commandToClient( %client, 'writeHint', "The signet ring now has some wax.");
    %client.inventory.state = "waxed";
  } else {
    commandToClient( %client, 'writeHint', "Ouch! It's too hot to touch.");
  }
}
```

Ο παραπάνω κώδικας βρίσκεται στο πρώτο μέρος του αρχείου desk.cs, το δεύτερο μέρος του οποίου αναλύθηκε στο τμήμα για τα **Containers**. Η ομαδοποίηση των παραπάνω αντικειμένων σαν «είδη γραφείου» οφείλεται στο γεγονός ότι αποτελούνται από μικρά αντικείμενα που αλληλεπιδρούν μεταξύ τους και βρίσκονται όλα στο ίδιο δωμάτιο.

Το αρχείο Desk.cs περιέχει περιγραφές των DataBlocks διαφόρων αλληλεπιδραστικών αντικειμένων, πολλά από τα οποία μπορεί να πάρει ο χρήστης στο inventory του. Η αλληλεπίδραση όλων των αντικειμένων γίνεται με σκοπό τη δημιουργία ενός σφραγισμένου φακέλου που περιέχει μια εξουσιοδότηση (αυτό είναι ένα αντικείμενο που θα δοθεί στον NPC ξεναγό του μουσείου για να ξεκινήσει την ξενάγηση). Χωρίς η σειρά να είναι απαραίτητα αυτή που αναφέρουμε, ο χρήστης πρέπει να διαβάσει ένα βιβλίο από τη βιβλιοθήκη που περιέχει πληροφορίες ως προς το πώς να γράψει μια εξουσιοδότηση, να πάρει το φτερό, να το βουτήξει στο μελανοδοχείο, να γράψει σ' ένα λευκό χαρτί την εξουσιοδότηση, να πάρει ένα φάκελο από το συρτάρι του γραφείου, να βάλει το γραμμένο

χαρτί στο φάκελο, να βρει το κλειδί που ξεκλειδώνει το σεντούκι, να ανοίξει το σεντούκι και να πάρει από μέσα το δαχτυλίδι με το σύμβολο του μουσείου, να πάρει λίγο κερί και να βουλώσει τον φάκελο. Κατόπιν μπορεί να πάρει τον φάκελο και να τον δώσει στον ξεναγό. Έχουμε λοιπόν με τη σειρά που καταχωρούνται στο Desk.cs τις εξής κλάσεις αντικειμένων:

- **Inkquill:** Το φτερό είναι ένα αντικείμενο που μπορεί ο χρήστης να τοποθετήσει στο inventory του. Έχει την μεταβλητή state που μπορεί να πάρει τιμές "inked" και "uninked" (ξεκινά με "uninked"). Αυτή η μεταβλητή εκφράζει το αν το φτερό έχει μελάνι, και άρα μπορεί ο χρήστης να γράψει με αυτό. Η μεταβλητή αλλάζει σε "inked" μόνο με χρήση του φτερού στο μελανοδοχείο. Ο τρόπος με τον οποίο τοποθετεί ο χρήστης ένα αντικείμενο στο inventory του αναλύθηκε στο σκέλος 6.1.
- **Inkwell:** Το μελανοδοχείο είναι ένα αλληλεπιδραστικό αντικείμενο. Όταν το ενεργοποιεί ο χρήστης και έχει στο inventory του το φτερό (κλάση Inkquill) το οποίο έχει μεταβλητή state "uninked" τότε η μεταβλητή state του φτερού γίνεται "inked" και μπορεί ο χρήστης να γράψει με αυτό. Αν το state του φτερού είναι ήδη "inked" ενημερώνεται ο χρήστης με εντολή WriteHint στον πελάτη του.
- **Paper:** Το λευκό χαρτί είναι ένα αλληλεπιδραστικό αντικείμενο. Αν ο χρήστης έχει στο inventory του το φτερό με αρκετό μελάνι (δηλαδή "inked" state) και έχει διαβάσει το βιβλίο με ID book_FormFilling (δηλαδή το state του παραπάνω αντικειμένου είναι "read" όπως αναλύσαμε στο σκέλος 6.5) τότε το αντικείμενο Paper απομακρύνεται από την εφαρμογή (με την εντολή delete) και στο σημείο που βρισκόταν τοποθετείται ένα νέο αντικείμενο (με χρήση της new) της κλάσης Paper_Written. Αν ο χρήστης δεν έχει διαβάσει το βιβλίο book_FormFilling ή έχει άλλα αντικείμενα στο Inventory του τότε ο χρήστης ενημερώνεται με εντολή WriteHint στον πελάτη του. Σε όλες τις άλλες περιπτώσεις τοποθετείται στο στοιχείο PaintingCover του GUI η εικόνα που σώζεται στη μεταβλητή page και ακολουθείται η ίδια διαδικασία με αυτή στα βιβλία και στους πίνακες που αναπτύχθηκε στο σκέλος 6.5.
- **Paper_Written:** Το γραμμένο χαρτί είναι ένα αλληλεπιδραστικό αντικείμενο. Αν ο χρήστης έχει στο inventory του έναν άδειο φάκελο (κλάση Envelope_Empty) τότε το αντικείμενο Paper_Written απομακρύνεται από την εφαρμογή (με την εντολή delete) και στο σημείο που βρισκόταν τοποθετείται ένα νέο αντικείμενο (με χρήση της new) της κλάσης Envelope. Αν ο χρήστης έχει άλλα αντικείμενα στο Inventory του τότε ο χρήστης ενημερώνεται με εντολή WriteHint στον πελάτη του. Σε όλες τις άλλες περιπτώσεις τοποθετείται στο στοιχείο PaintingCover του GUI η εικόνα που σώζεται στη μεταβλητή page και ακολουθείται η ίδια διαδικασία με αυτή στα βιβλία και στους πίνακες που αναπτύχθηκε στο σκέλος 6.5.
- **Envelope_Empty:** Ο άδειος φάκελος είναι ένα αντικείμενο που μπορεί ο χρήστης να τοποθετήσει στο inventory του. Αν χρησιμοποιηθεί σε αντικείμενο της κλάσης Paper_Written τότε δημιουργείται ένα αντικείμενο της κλάσης Envelope στη θέση του Paper_Written ενώ ταυτόχρονα απομακρύνεται το Envelope_Empty από το Inventory του χρήστη. Ο τρόπος με τον οποίο τοποθετεί ο χρήστης ένα αντικείμενο στο inventory του αναλύθηκε στο σκέλος 6.1.
- **Envelope:** Ο φάκελος με την εξουσιοδότηση είναι ένα αλληλεπιδραστικό αντικείμενο. Αν ο χρήστης έχει στο inventory το δαχτυλίδι (αντικείμενο της κλάσης Signet_Ring) και το δαχτυλίδι έχει state "waxed" τότε ο φάκελος απομακρύνεται από την εφαρμογή (με την εντολή delete) και στο σημείο που βρισκόταν τοποθετείται ένα νέο αντικείμενο (με χρήση της new) της κλάσης Envelope_Sealed.
- **Envelope_Sealed:** Ο βουλωμένος φάκελος είναι ένα αντικείμενο που μπορεί ο χρήστης να τοποθετήσει στο inventory του και να τον δώσει στον NPC ξεναγό. Ο τρόπος με τον οποίο τοποθετεί ο χρήστης ένα αντικείμενο στο inventory του αναλύθηκε στο σκέλος 6.1.
- **Signet_Ring:** Το δαχτυλίδι με τη σφραγίδα του μουσείου είναι ένα αντικείμενο που μπορεί ο χρήστης να τοποθετήσει στο inventory του. Το δαχτυλίδι αυτό έχει την μεταβλητή state που μπορεί να πάρει την τιμή "waxed" (ξεκινά με κενή τιμή). Το

state αλλάζει σε “waxed” μόνο αν χρησιμοποιηθεί το δαχτυλίδι στο κερί. Το δαχτυλίδι με “waxed” state χρησιμοποιείται πάνω στο Envelope για να δημιουργήσει το Envelope_Sealed. Ο τρόπος με τον οποίο παίρνει ο χρήστης ένα αντικείμενο στο inventory του αναλύθηκε στο σκέλος 6.1.

- **Candle:** Το κερί είναι ένα αλληλεπιδραστικό αντικείμενο. Αν ο χρήστης έχει στο inventory του αντικείμενο της κλάσης Signet_Ring τότε αλλάζει το state του δαχτυλιδιού σε “waxed” και έτσι μπορεί να σφραγίσει το φάκελο. Σε όλες τις άλλες περιπτώσεις ενημερώνεται ο χρήστης με εντολή WriteHint στον πελάτη του.

## 6.7: Εκθέματα (Display_item.cs)

```
datablock StaticShapeData(DB_showbox){
  category = "Display_Item";
  shapefile = "~/data/furniture/showbox.dts";
  className = "DItem";
};

function DItem::onAdd(%this,%obj){}
function DItem::onCollision(%this,%obj,%col,%vec,%speed){}
function DItem::onRemove(%this, %obj){}

function DItem::FreeLook(%this, %obj, %client){
  %conn = %client.getControlObject();
  if (%conn == %client.player){
    %transform = %obj.getTransform();
    %conn = %client.camera;
    %conn.target = %obj.getID();
    %conn.mode = "Observe";
    //%conn.setOrbitMode(%obj, %transform,0.5,1,5);
    Echo("Resetting Camera's Xform");
    %conn.setOrbitMode(%obj, "0 0 0 1 0 0 90",0.75,1.5,1);
    %client.setControlObject(%conn);
  }
}

function DItem::Look(%this, %client, %direction){
  Echo("Now we change the camera's position");
  %camera_xform = %client.camera.getTransform();
  Echo("Camera's previous Xform is " @ %camera_xform );
  if( %direction $= "up" ){
    %var_x = 0.1745;
    %var_y = 0.1745;
    %var_z = 0;
  } else if( %direction $= "down" ){
    %var_x = -0.1745;
    %var_y = -0.1745;
    %var_z = 0;
  } else if( %direction $= "right" ){
    %var_x = 0;
    %var_y = 0;
    %var_z = 0.1745;
  } else if( %direction $= "left" ){
    %var_x = 0;
    %var_y = 0;
    %var_z = -0.1745;
  }
  %t_x = GetWord(%camera_xform,3);
  %t_y = GetWord(%camera_xform,4);
  %t_z = GetWord(%camera_xform,5);
  %turn = GetWord(%camera_xform,6);
  Echo( "(t_x, t_y, t_z) = " @ %t_x SPC %t_y SPC %t_z );
  Echo( "turn = " @ %turn );
  //%turn = ( %turn / 3.1416 )*360;
  //Echo( "turn = " @ %turn @ " degrees" );
  %h_x = %t_x * %turn + %var_x;
  %h_y = %t_y * %turn + %var_y;
  %h_z = %t_z * %turn + %var_z;
  %new_turn = mSqrt( mPow( %h_x, 2 ) + mPow( %h_y, 2 ) + mPow( %h_z, 2 ) );
  Echo( "new_turn is " @ %new_turn );
  if( %new_turn < 0 ) {
    %new_turn = -%new_turn;
    %h_x = -%h_x;
    %h_y = -%h_y;
    %h_z = -%h_z;
    Echo("Changed the turn to " @ %new_turn );
  }
  %a = %h_x / %new_turn;
  %b = %h_y / %new_turn;
  %c = %h_z / %new_turn;
```

```
  Echo( "(a,b,c) = " @ %a SPC %b SPC %c );
  %new_camera_xform = GetWords(%camera_xform,0,2) SPC %a SPC %b SPC %c SPC %new_turn;
  Echo("Camera's new Xform is " @ %new_camera_xform );
  %client.camera.setTransform(%new_camera_xform);
  %camera_xform = %client.camera.getTransform();
  Echo("Camera's resulting Xform is " @ %camera_xform );
}
```

Τα εκθέματα (μακέτες) της εφαρμογής είναι αντικείμενα τύπου TSStatic και βρίσκονται μέσα σε ξύλινες κατασκευές που είναι αλληλεπιδραστικό και ανήκουν στην κλάση DItem. Όταν ο χρήστης χρησιμοποιεί ένα αντικείμενο της κλάσης DItem, ο χρήστης χάνει τον έλεγχο του avatar και η κάμερα της εφαρμογής τίθεται σε μια θέση παρατήρησης του εκθέματος. Ταυτόχρονα εμφανίζεται ο cursor και μερικά κουμπιά στο αριστερό άκρο της οθόνης με τα οποία μπορεί ο χρήστης να μετακινήσει την κάμερα όπως θέλει (πάνω, κάτω, αριστερά ή δεξιά) για να μπορεί να εξετάσει το έκθεμα. Πάτημα κάθε κουμπιού καλεί την εντολή Look (μέσω commandToServer από την πλευρά του πελάτη) με διαφορετικό όρισμα direction. Ανάλογα το direction, αλλάζει η θέση της κάμερας έτσι ώστε να παραμένει πάντα σε σταθερή απόσταση από το έκθεμα. Επίσης υπάρχει ένα κουμπί FreeLook το οποίο καλεί την συνάρτηση FreeLook η οποία θέτει την κάμερα στον απόλυτο έλεγχο του χρήστη: με κέντρο το έκθεμα και μοναδιαία ακτίνα, ο χρήστης μπορεί να μετακινεί την κάμερα όπως θέλει με κίνηση του ποντικιού. Αυτή η λειτουργία είναι εφικτή με χρήση της setOrbitMode. Ο χρήστης μπορεί να βγει από την διαδικασία εξέτασης του αντικειμένου και να επανακτήσει τον έλεγχο του avatar με πάτημα του πλήκτρου Q για να ενεργοποιηθεί η εντολή “QuitOrbit” στον εξυπηρετητή (όπως αναλύσαμε στο σκέλος 4.4).

## 6.8: Εφέ με φώτα (Magic_light.cs)

```
datablock StaticShapeData(DB_Mbutton1){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button1.dts";
};

datablock StaticShapeData(DB_Mbutton2){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button2.dts";
};

datablock StaticShapeData(DB_Mbutton3){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button3.dts";
};

datablock StaticShapeData(DB_Mbutton4){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button4.dts";
};

datablock StaticShapeData(DB_Mbutton5){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button5.dts";
};

datablock StaticShapeData(DB_Mbutton6){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button6.dts";
};

datablock StaticShapeData(DB_Mbutton7){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button2.dts";
};

datablock StaticShapeData(DB_Mbutton8){
  className = "MButton";
  category = "Magic_Button";
  shapefile = "~/data/active/podium/button_set1/magic_button4.dts";
};

// +-----------------------------------+
// |   ACTIVATE                        |
// +-----------------------------------+


//function MButton::Activate(%this, %obj, %client){
//  if(%obj.state $= "on"){                                          // if it is on, switch
it off...
//    Echo("Switching off" @ %obj.getID() @ "!");
//    %obj.playThread(0,"on");
//    %obj.state = "switch_off";
//    %this.schedule(1000, "changeState", %obj);
//    if( %obj.getID() == magic_button1.getID() ){
//      $current_effect = "DeleteAll";
//      %this.deleteAll(%obj, %client, 0);
//    }
//  }
//  if(%obj.state $= "off"){                                         // if it is off,
switch it on...
//    Echo("Switching on " @ %obj.getID() @ "!");
```

```
//    %obj.playThread(0,"off");
//    %obj.state = "switch_on";
//    %this.schedule(1000, "changeState", %obj);
//    %this.SpawnLight(%obj, %client);
//  }
//}

function MButton::Activate(%this, %obj, %client){
  Echo("Press the " @ %obj.getID() @ "!");
  %obj.playThread(0,"on");
  %obj.state = "switch_off";
  %this.schedule(1000, "changeState", %obj);
  if( %obj.getID() == magic_button1.getID() ){
    if( $current_effect $= "AllDeleted" ){
      Echo("MoveToCenter!");
      %this.SpawnLight(%obj, %client);
    } else {
      $current_effect = "DeleteAll";
      %this.deleteAll(%obj, %client, 0);
    }
  } else {
    if( $current_effect $= "TrailDeleted" ){
      Echo("New effect!");
      %this.SpawnLight(%obj, %client);
    } else {
      commandToClient( %client, 'writeHint', "You must first reset the light before
making a new effect." );
    }
  }
}

function MButton::changeState(%this, %obj){
  Error("Button Usable!");
//  if( %obj.state $= "switch_on"){
//    %obj.state = "on";
//  }
  if( %obj.state $= "switch_off"){
    %obj.state = "off";
  }
}

// SPECIFIC BUTTON LIGHT EFFECTS

// +-----------------------------------+
// |   SPAWN LIGHT                     |
// +-----------------------------------+


function MButton::SpawnLight(%this, %obj, %client){
  if( $current_effect $= "TrailDeleted" ){
    Error("Skip MoveTo!");
    %this.Schedule(100, "nextState", %obj, %i);
  } else {
//  %firename = %obj.getName() @ "_fire";
    %firename = "magic_fire";
    if( %firename.getID() ){
      Echo("Light exists! Deleting...");
      %firename.delete();
    }
    Echo("Spawning new Light!");
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
    %fire = new ParticleEmitterNode("magic_fire") {
      scale = "1 1 1";
      dataBlock = "MagicLight1ParticleEmitterNode";
      emitter = "MagicLight1ParticleEmitter";
      velocity = "1";
    };
    %fire.setTransform($light_top SPC "1 0 0 0");
    $currentEffect = "MoveToCenter";
    %this.Schedule(100, "move", %obj, $light_center, 0, $current_effect);
  }
```

```
}

// +-----------------------------------+
// |   MOVE                            |
// +-----------------------------------+

function MButton::move(%this, %obj, %target, %i, %effect){
  Echo("Current effect = " @ $current_effect @ ", effect = " @ %effect );
  if( $current_effect !$= %effect ){
    Error("STOPIT!");
    return;
  }

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();
  %finished = 0;

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

//  %old_trail_name = %obj.getName() @ "_trail" @ %i;
  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
//  %trail = new ParticleEmitterNode(%obj.getName() @ "_trail" @ %i) {
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS

  %sub_vector = VectorSub( %fire_id.getPosition(), %target );
  %sub_vector = VectorScale( %sub_vector, 0.5 );
  while( VectorLen( %sub_vector ) > $light_maxDistance ){
    %sub_vector = VectorScale( %sub_vector, 0.5 );
  }
  if( VectorLen( %sub_vector ) < $light_minDistance ){
    %fire_xform = %target;
    %finished = 1;
  } else {
    Echo("SubVector = " @ %sub_vector);
    %fire_xform = VectorSub( %fire_id.getPosition(), %sub_vector );
//    %fire_xform = VectorAdd( %fire_id.getPosition(), %sub_vector );
    Echo("fire_xform = " @ %fire_xform);
  }

  //%fire_xform = VectorSub( %fire_id.getPosition(), %target );
  %fire_id.delete();
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };
  %fire.setTransform(%fire_xform SPC "1 0 0 0");
  if( %finished == 1 ){
```

```
    Error("FINISHED!");
    %this.Schedule(100, "nextState", %obj, %i);
  } else {
    %this.Schedule(100, "move", %obj, %target, %i, %effect);
  }
}

// +-----------------------------------+
// |   DELETE TRAIL                    |
// +-----------------------------------+

function MButton::deleteTrail(%this, %obj, %i){
  Echo("Current effect = " @ $current_effect );
  Error("Delete Trail!");

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

//  %old_trail_name = %obj.getName() @ "_trail" @ %i;
  %old_trail_name = "trail" @ %i;
  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
    %this.Schedule(100, "deleteTrail", %obj, %i);
  } else {
    $current_effect = "TrailDeleted";
  }
}

// +-----------------------------------+
// |   DELETE ALL                      |
// +-----------------------------------+

function MButton::deleteAll(%this, %obj, %i){
  Echo("Current effect = " @ $current_effect );
  Error("Delete All!");

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

//  %old_trail_name = %obj.getName() @ "_trail" @ %i;
  %old_trail_name = "trail" @ %i;
  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
    %this.Schedule(100, "deleteAll", %obj, %i);
  } else {
    %firename = "magic_fire";
    if( %firename.getID() ){
      %firename.delete();
    }
    $current_effect = "AllDeleted";
    Echo("DeleteAll function ended.");
  }
}

// +-----------------------------------+
// |   WHIRL                           |
// +-----------------------------------+

function MButton::whirl(%this, %obj, %i, %r, %theta){
```

```
  Echo("Current effect = " @ $current_effect );
  if( $current_effect !$= "Whirl" ){
    Error("STOPIT!");
    return;
  }

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
//    %old_trail_name = %obj.getName() @ "_trail0";
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
//  %trail = new ParticleEmitterNode(%obj.getName() @ "_trail" @ %i) {
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS

//  %fire_x = GetWord($light_center,0) + %r*mSin(%theta);
//  %fire_y = GetWord($light_center,1) + %r*mCos(%theta);
//  %fire_z = GetWord($light_center,2);

  %fire_x = GetWord($light_center,0) + %r*mCos(%theta);
  %fire_y = GetWord($light_center,1);
  %fire_z = GetWord($light_center,2) + %r*mSin(%theta);

  Echo("#Theta = " @ %theta);
  Echo("* Sin = " @ mSin(%theta));
  Echo("* Cos = " @ mCos(%theta));

  %theta = %theta + 0.0628;
  if(%theta >= 6.28){
    %theta = 0;
  }
  if( %r < $r_max ){
    // scalable radius ADDITION
    if( %r < $r_max/4 ){
      %r = %r + 0.1;
      Echo("Increment 0.1");
    } else if ( %r < $r_max/2 ){
      %r = %r + 0.05;
      Echo("Increment 0.05");
    } else {
      %r = %r + 0.01;
      Echo("Increment 0.01");
    }
    Error("More");
  }

  %fire_id.delete();
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
```

```
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };

  %fire_xform = %fire_x SPC %fire_y SPC %fire_z;
  Echo("Fire_Xform = " @ %fire_xform);
  %fire.setTransform(%fire_xform SPC "1 0 0 0");
  //%fire.setTransform(%fire_x SPC %fire_y SPC %fire_z SPC "1 0 0 0");
  %this.Schedule(100, "whirl", %obj, %i, %r, %theta);
}

// +-----------------------------------+
// |   OSCILLATE                       |
// +-----------------------------------+

function MButton::oscillate(%this, %obj, %i, %asc_flag){
  Echo("Current effect = " @ $current_effect );
  if( $current_effect !$= "Oscillate" ){
    Error("STOPIT!");
    return;
  }

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();

  // ADDITIONS FOR TRAIL
//  if( %i > $max_trail_particles ){
  if( %i > 25 ){
//    %old_trail_name = %obj.getName() @ "_trail0";
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
//  %trail = new ParticleEmitterNode(%obj.getName() @ "_trail" @ %i) {
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS

  %fire_z = GetWord(%fire_id.getPosition(),2);
  if( ( %fire_z > GetWord($light_center,2) + $oscillation_limit ) && ( %asc_flag == 1 )
){
    %asc_flag = 0;
  }
  if( ( %fire_z < GetWord($light_center,2) - $oscillation_limit ) && ( %asc_flag == 0 )
){
    %asc_flag = 1;
  }

  if( %asc_flag == 1 ){
    %fire_z = %fire_z + 0.5;
  } else {
    %fire_z = %fire_z - 0.5;
  }
```

```
  %fire_id.delete();
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };

  %fire_xform = GetWord($light_center,0) SPC GetWord($light_center,1) SPC %fire_z;
  Echo("Fire_Xform = " @ %fire_xform);
  %fire.setTransform( %fire_xform SPC "1 0 0 0");
  %this.Schedule(100, "oscillate", %obj, %i, %asc_flag);
}

// +-----------------------------------+
// |    RANDOM                         |
// +-----------------------------------+

function MButton::random(%this, %obj, %i, %temp_Xform){
  Echo("Current effect = " @ $current_effect );
  if( $current_effect !$= "Random" ){
    Error("STOPIT!");
    return;
  }

  Error("Random!");

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();

  // ADDITIONS FOR TRAIL
//  if( %i > $max_trail_particles ){
  if( %i > 25 ){
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
//  %trail = new ParticleEmitterNode(%obj.getName() @ "_trail" @ %i) {
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS

  %fire_Xform = %fire_id.getPosition();
  %fire_x = GetWord(%fire_Xform,0);
  %fire_y = GetWord(%fire_Xform,1);
  %fire_z = GetWord(%fire_Xform,2);

  %fire_newXform = VectorAdd( %fire_Xform, %temp_xform );

  while( VectorLen( VectorDist( %fire_newXform, $light_center ) ) > $r_max ){
    %temp_x = 0.02*( getrandom(20)-10 );
    %temp_y = 0.02*( getrandom(20)-10 );
    %temp_z = 0.02*( getrandom(20)-10 );
```

```
    %temp_Xform = %temp_x SPC %temp_y SPC %temp_z;
    %fire_newXform = VectorAdd( %fire_Xform, %temp_Xform );
  }

  Echo( "fire_newXform = " @ %fire_newXform );
  Echo( "temp_Xform = " @ %temp_Xform );

  %fire_id.delete();
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };

  %fire.setTransform( %fire_newXform SPC "1 0 0 0");
  %this.Schedule(100, "random", %obj, %i, %temp_Xform);
}

// +-----------------------------------+
// |   SPIRAL                          |
// +-----------------------------------+

function MButton::spiral(%this, %obj, %i, %r, %theta, %asc_flag){
  Echo("Current effect = " @ $current_effect );
  if( $current_effect !$= "Spiral" ){
    Error("STOPIT!");
    return;
  }

  Error("Spiral!");

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
//    %old_trail_name = %obj.getName() @ "_trail0";
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
//  %trail = new ParticleEmitterNode(%obj.getName() @ "_trail" @ %i) {
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS

  %fire_x = GetWord($light_center,0) + %r*mSin(%theta);
  %fire_y = GetWord($light_center,1) + %r*mCos(%theta);

  %fire_z = GetWord(%fire_id.getPosition(),2);
  if( ( %fire_z > GetWord($light_center,2) + $oscillation_limit ) && ( %asc_flag == 1 )
){
    %asc_flag = 0;
```

```
  }
  if( ( %fire_z < GetWord($light_center,2) - $oscillation_limit ) && ( %asc_flag == 0 )
){
    %asc_flag = 1;
  }

  Echo("#Theta = " @ %theta);
  Echo("* Sin = " @ mSin(%theta));
  Echo("* Cos = " @ mCos(%theta));

  %theta = %theta + 0.062;
  if(%theta >= 6.28){
    %theta = 0;
  }
  if( %r < $r_max ){
    // scalable radius ADDITION
    if( %r < $r_max/4 ){
      %r = %r + 0.1;
      Echo("Increment 0.1");
    } else if ( %r < $r_max/2 ){
      %r = %r + 0.05;
      Echo("Increment 0.05");
    } else {
      %r = %r + 0.01;
      Echo("Increment 0.01");
    }
    Error("More");
  } else {
    // start spiraling only at maximum radius
    if( %asc_flag == 1 ){
      %fire_z = %fire_z + 0.03;
    } else {
      %fire_z = %fire_z - 0.03;
    }
  }

  %fire_id.delete();
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };

  %fire_xform = %fire_x SPC %fire_y SPC %fire_z;
  Echo("Fire_Xform = " @ %fire_xform);
  %fire.setTransform(%fire_xform SPC "1 0 0 0");
  //%fire.setTransform(%fire_x SPC %fire_y SPC %fire_z SPC "1 0 0 0");
  %this.Schedule(100, "spiral", %obj, %i, %r, %theta, %asc_flag);
}

// +-----------------------------------+
// |   RAD_SPIRAL                      |
// +-----------------------------------+

function MButton::rad_spiral(%this, %obj, %i, %r, %theta, %asc_flag){
  Echo("Current effect = " @ $current_effect );
  if( $current_effect !$= "Rad_Spiral" ){
    Error("STOPIT!");
    return;
  }

  Error("Rad_Spiral!");

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
```

```
//    %old_trail_name = %obj.getName() @ "_trail0";
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
//  %trail = new ParticleEmitterNode(%obj.getName() @ "_trail" @ %i) {
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS


  %fire_x = GetWord($light_center,0) + mAbs(%r)*mSin(%theta);
  %fire_y = GetWord($light_center,1) + mAbs(%r)*mCos(%theta);

  %fire_z = GetWord(%fire_id.getPosition(),2);
  if( ( %fire_z > GetWord($light_center,2) + $oscillation_limit ) && ( %asc_flag == 1 )
){
    %asc_flag = 0;
  }
  if( ( %fire_z < GetWord($light_center,2) - $oscillation_limit ) && ( %asc_flag == 0 )
){
    %asc_flag = 1;
  }

  if( %asc_flag == 1 ){
    %fire_z = %fire_z + 0.1;
  } else {
    %fire_z = %fire_z - 0.1;
  }

  Echo("#Theta = " @ %theta);
  Echo("* Sin = " @ mSin(%theta));
  Echo("* Cos = " @ mCos(%theta));

  %theta = %theta + 0.062;           //peripou 50 runs gia enan pliri kyklo
  if(%theta >= 6.28){
    %theta = 0;
  }
  if( %asc_flag == 1 ){
    if( %fire_z < GetWord($light_center,2) ){
      %r = %r + 0.03;
    } else {
      %r = %r - 0.03;
    }
  } else {
    if( %fire_z > GetWord($light_center,2) ){
      %r = %r + 0.03;
    } else {
      %r = %r - 0.03;
    }
  }

  Echo("#r = " @ %r);
  Echo("#z = " @ %fire_z);

  %fire_id.delete();
```

```
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };

  %fire_xform = %fire_x SPC %fire_y SPC %fire_z;
  Echo("Fire_Xform = " @ %fire_xform);
  %fire.setTransform(%fire_xform SPC "1 0 0 0");
  //%fire.setTransform(%fire_x SPC %fire_y SPC %fire_z SPC "1 0 0 0");
  %this.Schedule(100, "rad_spiral", %obj, %i, %r, %theta, %asc_flag);
}

// +-----------------------------------+
// |   GLOBE_SPIRAL                    |
// +-----------------------------------+

function MButton::globe_spiral(%this, %obj, %i, %theta, %asc_flag){
  Echo("Current effect = " @ $current_effect );
  if( $current_effect !$= "Globe_Spiral" ){
    Error("STOPIT!");
    return;
  }

  Error("Globe_Spiral!");

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
//    %old_trail_name = %obj.getName() @ "_trail0";
    %i = 0;
  } else {
    %i = %i + 1 ;
  }

  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                            // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS

  if( %asc_flag == 1 ){
    %h = 0.02;
  } else {
    %h = -0.02;
  }

  %fire_z = GetWord(%fire_id.getPosition(),2);
  %fire_z = %fire_z + %h;

  %dz = %fire_z - GetWord($light_center,2);
  if( %dz > $r_max ){
    %dz = $r_max;
  }
```

```
  if( %dz < -$r_max ){
    %dz = -$r_max;
  }
  %fi = mAsin( %dz / $r_max );

  Echo(" h / r_max = " @ ( %fire_z - GetWord($light_center,2) ) / $r_max );
  Echo(" fi = " @ %fi );
  %r = $r_max*mCos(%fi);
  Echo(" r = " @ %r );

  %fire_x = GetWord($light_center,0) + mAbs(%r)*mSin(%theta);
  %fire_y = GetWord($light_center,1) + mAbs(%r)*mCos(%theta);

  if( ( %fire_z > GetWord($light_center,2) + $r_max ) && ( %asc_flag == 1 ) ){
    %asc_flag = 0;
  }
  if( ( %fire_z < GetWord($light_center,2) - $r_max ) && ( %asc_flag == 0 ) ){
    %asc_flag = 1;
  }

  Echo("#Theta = " @ %theta);
  Echo("* Sin = " @ mSin(%theta));
  Echo("* Cos = " @ mCos(%theta));

  %theta = %theta + 0.062;           //peripou 50 runs gia enan pliri kyklo
  if(%theta >= 6.28){
    %theta = 0;
  }

  Echo("#r = " @ %r);
  Echo("#z = " @ %fire_z);

  %fire_id.delete();
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };

  %fire_xform = %fire_x SPC %fire_y SPC %fire_z;
  Echo("Fire_Xform = " @ %fire_xform);
  %fire.setTransform(%fire_xform SPC "1 0 0 0");
  //%fire.setTransform(%fire_x SPC %fire_y SPC %fire_z SPC "1 0 0 0");
  %this.Schedule(100, "globe_spiral", %obj, %i, %theta, %asc_flag);
}

// +-----------------------------------+
// |   ASTERISK                        |
// +-----------------------------------+

function MButton::asterisk(%this, %obj, %i, %theta, %asc_flag){
  Echo("Current effect = " @ $current_effect );
  if( $current_effect !$= "Asterisk" ){
    Error("STOPIT!");
    return;
  }

  Echo("Theta = " @ %theta );

//  %firename = %obj.getName() @ "_fire";
  %firename = "magic_fire";
  %fire_id = %firename.getID();

  // ADDITIONS FOR TRAIL
  if( %i > $max_trail_particles ){
//    %old_trail_name = %obj.getName() @ "_trail0";
    %i = 0;
  } else {
    %i = %i + 1 ;
```

```
  }

  %old_trail_name = "trail" @ %i;

  if( %old_trail_name.getID() ){                          // EXISTS
    %old_trail_id = %old_trail_name.getID();
    %old_trail_id.delete();
    Error( %old_trail_name SPC %old_trail_id SPC "deleted." );
  }

  %trail_xform = %fire_id.getPosition();
//  %trail = new ParticleEmitterNode(%obj.getName() @ "_trail" @ %i) {
  %trail = new ParticleEmitterNode("trail" @ %i) {
    scale = "1 1 1";
    dataBlock = "Trail1ParticleEmitterNode";
    emitter = "Trail1ParticleEmitter";
    velocity = "1";
  };
  %trail.setTransform(%trail_xform SPC "1 0 0 0");
  // END OF TRAIL ADDITIONS

  %fire_x = GetWord(%fire_id.getPosition(),0);
  %fire_z = GetWord(%fire_id.getPosition(),2);

  %oscillation_limit_x = $oscillation_limit*mCos(%theta);
  %oscillation_limit_z = $oscillation_limit*mSin(%theta);

  Echo("# x = " @ %oscillation_limit_x);
  Echo("# z = " @ %oscillation_limit_z);

  if( %asc_flag == 1 ){
    if( ( %fire_x > GetWord($light_center,0) + %oscillation_limit_x ) || ( %fire_z >
GetWord($light_center,2) + %oscillation_limit_z ) ){
      %asc_flag = 0;
      Error("Change Flag! DESCEND!");
    }
  } else if( %asc_flag == 0 ){
    if( ( %fire_x < GetWord($light_center,0) - %oscillation_limit_x ) || ( %fire_z <
GetWord($light_center,2) - %oscillation_limit_z ) ){
      %asc_flag = 1;
      Error("Change Flag! ASCEND!");
    }
  }

  if( %asc_flag == 1 ){
    %fire_x = %fire_x + 0.5*mCos(%theta);
    %fire_z = %fire_z + 0.5*mSin(%theta);
  } else {
    %fire_x = %fire_x - 0.5*mCos(%theta);
    %fire_z = %fire_z - 0.5*mSin(%theta);
  }

  if( %asc_flag == 1 ){
    if( ( %fire_z == GetWord($light_center,2) ) && ( %fire_x == GetWord($light_center,0)
) ){
      Error("Increase Theta!");
      %theta = %theta + 30;
    }
  }

  %fire_id.delete();
//  %fire = new ParticleEmitterNode(%obj.getName() @ "_fire") {
  %fire = new ParticleEmitterNode("magic_fire") {
    scale = "1 1 1";
    dataBlock = "MagicLight1ParticleEmitterNode";
    emitter = "MagicLight1ParticleEmitter";
    velocity = "1";
  };

  %fire_xform = %fire_x SPC GetWord($light_center,1) SPC %fire_z;
  Echo("Fire_Xform = " @ %fire_xform);
```

```
  %fire.setTransform( %fire_xform SPC "1 0 0 0");
  %this.Schedule(100, "asterisk", %obj, %i, %theta, %asc_flag);
}

// +-----------------------------------+
// |   NEXTSTATE                       |
// +-----------------------------------+

function MButton::nextState(%this, %obj, %i){
  Echo("Next State");
  Echo("%obj = " @ %obj SPC %obj.getID() );
  if( %obj.getID() == magic_button1.getID() ){
    $current_effect = "DeleteTrail";
    %this.Schedule(100, "deleteTrail", %obj, %i);
  } else if( %obj.getID() == magic_button2.getID() ){
    $current_effect = "Whirl";
    %this.Schedule(100, "whirl", %obj, %i, 0, 0);
  } else if( %obj.getID() == magic_button4.getID() ){
    $current_effect = "Oscillate";
    %this.Schedule(100, "oscillate", %obj, %i, 1);
  } else if( %obj.getID() == magic_button3.getID() ){
    $current_effect = "Random";
    %this.Schedule(100, "random", %obj, %i, "0 0.2 0.2");
  } else if( %obj.getID() == magic_button6.getID() ){
    $current_effect = "Spiral";
    %this.Schedule(100, "spiral", %obj, %i, 0, 0, 0);
  } else if( %obj.getID() == magic_button5.getID() ){
    $current_effect = "Rad_Spiral";
    %this.Schedule(100, "rad_spiral", %obj, %i, 0, 0, 0);
  } else if( %obj.getID() == magic_button8.getID() ){
    if( $current_effect $= "Globe_Spiral_prepare" ){
      $current_effect = "Globe_Spiral";
      %this.Schedule(100, "globe_spiral", %obj, %i, 0, 0);
    } else if( $current_effect $= "TrailDeleted" ){
      $current_effect = "Globe_Spiral_prepare";
      %target_z = GetWord($light_center,2) + $r_max;
      %target = GetWords($light_center,0,1) SPC %target_z;
      %this.Schedule(100, "move", %obj, %target, 0, $current_effect);
    }
  } else if( %obj.getID() == magic_button7.getID() ){
    if( $current_effect $= "Pentagram1" ){
      $current_effect = "Pentagram2";
      %target_x = GetWord($light_center,0) + $r_max*0.588;
      %target_z = GetWord($light_center,2) + $r_max*(-0.81);
      %target = %target_x SPC GetWord($light_center,1) SPC %target_z;
      %this.Schedule(100, "move", %obj, %target, %i, $current_effect);
    } else if( $current_effect $= "Pentagram2" ){
      $current_effect = "Pentagram3";
      %target_x = GetWord($light_center,0) + $r_max*(-0.951);
      %target_z = GetWord($light_center,2) + $r_max*0.31;
      %target = %target_x SPC GetWord($light_center,1) SPC %target_z;
      %this.Schedule(100, "move", %obj, %target, %i, $current_effect);
    } else if( $current_effect $= "Pentagram3" ){
      $current_effect = "Pentagram4";
      %target_x = GetWord($light_center,0) + $r_max*0.951;
      %target_z = GetWord($light_center,2) + $r_max*0.31;
      %target = %target_x SPC GetWord($light_center,1) SPC %target_z;
      %this.Schedule(100, "move", %obj, %target, %i, $current_effect);
    } else if( $current_effect $= "Pentagram4" ){
      $current_effect = "Pentagram5";
      %target_x = GetWord($light_center,0) + $r_max*(-0.588);
      %target_z = GetWord($light_center,2) + $r_max*(-0.81);
      %target = %target_x SPC GetWord($light_center,1) SPC %target_z;
      %this.Schedule(100, "move", %obj, %target, %i, $current_effect);
    } else if ( ( $current_effect $= "Pentagram5" ) || ( $current_effect $=
"TrailDeleted" ) ){
      if( $current_effect $= "TrailDeleted" ){
        %i = 0;
      }
      $current_effect = "Pentagram1";
      %target_x = GetWord($light_center,0);
```

```
      %target_z = GetWord($light_center,2) + $r_max;
      %target = %target_x SPC GetWord($light_center,1) SPC %target_z;
      %this.Schedule(100, "move", %obj, %target, %i, $current_effect);
    }

//    $current_effect = "Asterisk";
//    %this.Schedule(100, "asterisk", %obj, %i, 90, 1);

  }
  Error("Current Effect = " @ $current_effect);
}

// +-----------------------------------+
// |   DESCRIPTORS AND GLOBALS         |
// +-----------------------------------+

$current_effect = "AllDeleted";

$light_center = "0.2 -56.6 5";
$light_top = "0.2 -56.6 50";
$light_maxDistance = 0.5;
$light_minDistance = 0.1;
$light_interval = 100;
$r_max = 4;
$oscillation_limit = 10;
//$max_trail_particles = 100;
$max_trail_particles = 100;

// -50.6 limit1_y
// -62.6 limit2_y

datablock ParticleData( MagicLight1Particle ){
       textureName        = "~/data/particles/fire/fire";
       dragCoefficient    = 0.0;
       gravityCoefficient = 0.0;
       inheritedVelFactor = 0.00;
       useInvAlpha        = false;
       spinRandomMin      = -90.0;
       spinRandomMax      = 270.0;

       lifetimeMS         = 1500;
       lifetimeVarianceMS = 250;

       times[0] = 0.0;
       times[1] = 0.5;
       times[2] = 1.0;

       colors[0] = "0.0 0.0 0.8 0.8";
       colors[1] = "0.0 0.6 0.8 0.5";
       colors[2] = "0.0 0.5 0.5 0.5";

       sizes[0] = 0.25;
       sizes[1] = 0.5;
       sizes[2] = 0.75;
};

datablock ParticleEmitterData( MagicLight1ParticleEmitter ){
       particles = MagicLight1Particle;

       ejectionPeriodMS = 5;
       periodVarianceMS = 0;

       ejectionVelocity = 0.65;
       velocityVariance = 0.10;

       thetaMin = 0.0;
       thetaMax = 180.0;
};

datablock ParticleEmitterNodeData( MagicLight1ParticleEmitterNode ){
       timeMultiple = 1;
```

```
};

// --- TRAIL ---

datablock ParticleData( Trail1Particle ){
	textureName		= "control/data/particles/fire/fire";
	dragCoefficient	= 0.0;
	gravityCoefficient = 0.0;
	inheritedVelFactor = 0.00;
	useInvAlpha		= false;
	spinRandomMin	= -90.0;
	spinRandomMax	= 270.0;

	lifetimeMS		= 1000;
	lifetimeVarianceMS = 250;

	times[0] = 0.0;
	times[1] = 0.5;
	times[2] = 1.0;

	colors[0] = "0.0 0.0 0.8 0.8";
	colors[1] = "0.0 0.6 0.8 0.5";
	colors[2] = "0.0 0.5 0.5 0.5";

	sizes[0] = 0.25;
	sizes[1] = 0.5;
	sizes[2] = 0.75;
};

datablock ParticleEmitterData( Trail1ParticleEmitter ){
	particles = Trail1Particle;

	ejectionPeriodMS = 50;
	periodVarianceMS = 5;

	ejectionVelocity = 0.50;
	velocityVariance = 0.10;

	thetaMin = 0.0;
	thetaMax = 180.0;
};

datablock ParticleEmitterNodeData( Trail1ParticleEmitterNode )
{
	timeMultiple = 1;
};
```

Τα εφέ με τα φώτα γίνονται σε ένα ειδικά διαμορφωμένο δωμάτιο και ελέγχονται από ένα πάνελ με αλληλεπιδραστικά κουμπιά. Κάθε κουμπί παράγει ένα διαφορετικό εφέ το οποίο σηματοδοτείται και από μια αλλαγή της παγκόσμιας μεταβλητής Current_Effect. Τα κινούμενα φώτα προσωμοιώνονται με δημιουργία στατικών πηγών particles σε μια συνεχόμενη τροχιά, με ταυτόχρονη καταστροφή προηγούμενων πηγών. Το πρώτο φως είναι τύπου MagicLight1ParticleEmitter ενώ αυτά που το ακολουθούν (διαγράφοντας την πορεία του) είναι τύπο Trail1ParticleEmittter και είναι σημαντικά μικρότερα: όταν αναφερόμαστε στο «φως» από εδώ και πέρα εννοούμε το MagicLight1ParticleEmitter. Κάθε εφέ ξεκινά με μεταφορά του φωτός από την κορυφή του δωματίου ως το κέντρο του (απ' όπου ξεκινά κάθε εφέ). Η κίνηση αυτή γίνεται από αλλεπάλληλες κλήσεις της συνάρτησης move κάθε 1sec. Όταν το φως φτάσει στο κέντρο του δωματίου (που ορίζεται στην παγκόσμια μεταβλητή Light_Center) τότε καλείται η nextState η οποία με τη σειρά της καλεί μια συνάρτηση (και αλλάζει την μεταβλητή current_effect) ανάλογα με το κουμπί που πατιέται. Οι συναρτήσεις των εφέ είναι οι εξής:

- **DeleteTrail:** εξαφανίζει όλες τις πηγές particles τύπου Trail1ParticleEmitter. Αποτέλεσμα αυτής της συνάρτησης είναι να μένει μόνο το φως (magic_fire) στο κέντρο του δωματίου, οπότε και το current_effect αλλάζει σε "TrailDeleted".
- **DeleteAll:** όμοιο με το DeleteTrail μόνο που στο τέλος σβήνει και το magic_fire.

- **Whirl:** ξεκινά το φως να διαγράφει έναν κύκλο (με χρήση ημιτόνων και συνημιτόνων) του οποίου η ακτίνα αυξάνεται μέχρι μια μέγιστη τιμή.
- **Oscillate:** το φως ταλαντώνεται στον Z-άξονα με σημείο ισορροπίας το κέντρο του δωματίου.
- **Random:** το φως κινείται σε τυχαία κατεύθυνση μέχρι να φτάσει στα όρια μιας νοητής σφαίρας γύρω από το κέντρο του δωματίου, οπότε αλλάζει πορεία με τυχαίο τρόπο έως ότου και πάλι φτάσει στα όρια της σφαίρας.
- **Spiral:** ξεκινά όπως η συνάρτηση whirl, διαγράφοντας έναν κύκλο στο επίπεδο XY, αλλά όταν η ακτίνα φτάνει στη μέγιστη τιμή της τότε το φως ξεκινά μια ταλάντωση στον Z-άξονα. Αποτέλεσμα αυτής της κίνησης του φωτός σ' έναν κύκλο αλλά και στον άξονα Z είναι η δημιουργία ενός spiral το οποίο κινείται είτε ανοδικά είτε καθοδικά.
- **Rad_Spiral:** ένα παραπάνω βήμα από τη Spiral, η συνάρτηση αυτή μεταβάλλει και την ακτίνα του κύκλου που διαγράφει το φως από το 0 έως μια μέγιστη τιμή. Αποτέλεσμα όλων αυτών των μεταβολών είναι η δημιουργία ενός σχήματος σαν κλεψύδρα από την τροχιά του φωτός. Επίσης να επισημανθεί ότι η Rad_spiral ξεκινά αμέσως να κινείται και επί του άξονα Z, ενώ η Spiral πρώτα κινείται όπως η whirl μόνο στο επίπεδο XY.
- **Globe_Spiral:** μια διαφορετική προσέγγιση των δύο προηγούμενων, το φως διαγράφει έναν κύκλο στο επίπεδο XY ενώ ταλαντώνεται στον άξονα Z μεταξύ δύο σημείων. Σε κάθε αλλαγή της συνιστώσας Z υπολογίζεται η ακτίνα του κύκλου έτσι ώστε να βρίσκεται πάντα στην περιφέρεια μιας νοητής σφαίρας γύρω από το κέντρο του δωματίου. Προφανώς η ταλάντωση στον άξονα Z έχει κέντρο επίσης το κέντρο του δωματίου και όριο ταλάντωσης ίσο με την ακτίνα της σφαίρας. Το εφέ αυτό ξεκινά από την κορυφή του κύκλου, στην οποία φτάνει το φως με μια κλήση της move.
- **Pentagram:** ουσιαστικά μια σειρά από αλλεπάλληλες move με διαφορετικούς στόχους, το φως κινείται συνεχόμενα από το ένα σημείο στο άλλο μεταξύ πέντε σημείων προκαθορισμένων στο δωμάτιο. Χάρη στην ύπαρξη του trail σχηματίζεται ένα γεωμετρικό αντικείμενο.

# ΚΕΦΑΛΑΙΟ 7:
# Τεχνητή Νοημοσύνη

## 7.1: Ορίζοντας το Bot και το animation του

```
datablock PlayerData( DB_Bot ){
  className = SampleBot;
  shapeFile = "~/data/players/genie/genie.dts";
  emap = true;
  renderFirstPerson = false;
  cameraMaxDist = 1;
  mass = 100;
  density = 10;
  drag = 0;
  maxdrag = 0;
  maxEnergy = 100;
  maxDamage = 100;
  maxForwardSpeed = 5;
  maxBackwardSpeed = 3;
  maxSideSpeed = 3;
  minJumpSpeed = 5;
  maxJumpSpeed = 10;
  runForce = 100000;
  jumpForce = 1000;
  runSurfaceAngle = 80;
  jumpSurfaceAngle = 80;
};
```

Για την περιγραφή ενός αντικειμένου PlayerData, είτε του avatar δηλαδή είτε του bot-ξεναγού, χρειάζονται πολλές μεταβλητές για να οριστεί η συμπεριφορά του. Μεταξύ αυτών αναγνωρίζουμε μερικές μεταβλητές που υπάρχουν και στα StaticShapes, όπως ο ορισμός της κλάση στην οποία ανήκει το DataBlock και το αρχείο στο οποίο βρίσκεται το μοντέλο που θα το απεικονίσει. Υπάρχουν όμως και πολλές άλλες μεταβλητές που ορίζουν τη μάζα του αντικειμένου, την ταχύτητά του, την μέγιστη γωνία την οποία μπορεί να ανέβει κτλ.

Καθώς για τα μοντέλα του avatar και του bot-ξεναγού δεν έχουμε ενσωματωμένα animations αλλά ξεχωριστά αρχεία τύπου DSQ για το animation τους, πρέπει να ορίσουμε ποια από αυτά τα αρχεία χρησιμοποιεί κάθε μοντέλο και σε ποιο όνομα animation αντιστοιχούν. Γι' αυτό υπάρχει αυτό το παρακάτω κομμάτι κώδικα στο αρχείο bot.cs

```
datablock TSShapeConstructor(GenieDts){
  baseShape = "~/data/players/genie/genie.dts";
  sequence0 = "~/data/players/genie/animations/idle.dsq root";
  sequence1 = "~/data/players/genie/animations/run.dsq run";
  sequence2 = "~/data/players/genie/animations/run.dsq back";
  sequence3 = "~/data/players/genie/animations/run.dsq side";
  sequence4 = "~/data/players/genie/animations/idle.dsq look";
  sequence5 = "~/data/players/genie/animations/idle.dsq head";
  sequence6 = "~/data/players/genie/animations/idle.dsq fall";
  sequence7 = "~/data/players/genie/animations/idle.dsq land";
  sequence8 = "~/data/players/genie/animations/idle.dsq jump";
  sequence9 = "~/data/players/genie/animations/idle.dsq death";
  sequence10 = "~/data/players/genie/animations/spellcast.dsq death2";     // custom
  sequence11 = "~/data/players/genie/animations/idle.dsq death3";          // custom
  sequence12 = "~/data/players/genie/animations/idle.dsq death4";          // custom
  sequence13 = "~/data/players/genie/animations/idle.dsq death5";          // custom
  sequence14 = "~/data/players/genie/animations/idle.dsq death6";          // custom
  sequence15 = "~/data/players/genie/animations/idle.dsq death7";
  sequence16 = "~/data/players/genie/animations/idle.dsq death8";
  sequence17 = "~/data/players/genie/animations/idle.dsq death9";
  sequence18 = "~/data/players/genie/animations/idle.dsq death10";
  sequence19 = "~/data/players/genie/animations/idle.dsq death11";
  sequence20 = "~/data/players/genie/animations/idle.dsq looksn";
```

```
   sequence21 = "~/data/players/genie/animations/idle.dsq lookms";
   sequence22 = "~/data/players/genie/animations/idle.dsq scoutroot";
   sequence23 = "~/data/players/genie/animations/idle.dsq headside";
   sequence24 = "~/data/players/genie/animations/idle.dsq light_recoil";
   sequence25 = "~/data/players/genie/animations/idle.dsq sitting";
   sequence26 = "~/data/players/genie/animations/idle.dsq celsalute";
   sequence27 = "~/data/players/genie/animations/idle.dsq celwave";
   sequence28 = "~/data/players/genie/animations/idle.dsq standjump";
   sequence29 = "~/data/players/genie/animations/idle.dsq looknw";
};
```

Παρατηρούμε ότι υπάρχουν πολλά sequences τα οποία ακολουθούνται από το αρχείο DSQ που τα περιγράφει και το όνομα στο οποίο αντιστοιχούν. Κάποια από αυτά τα sequences όπως το run, back, side κτλ. καλούνται απ' ευθείας από την μηχανή όταν το Bot κινείται. Αντίθετα, κάποια custom death animations καλούνται μέσα από τον κώδικα με playThread (τα death animations δεν είναι υποχρεωτικά animations θανάτου, μπορούν να επιτελέσουν πολλούς ρόλους όπως εδώ). Να παρατηρηθεί επίσης η ύπαρξη του baseshape στην αρχή του TSShapeConstructor, που ουσιαστικά ενσωματώνει όλα τα sequences στο αρχείο DTS. Έτσι είναι εφικτό να ορίσουμε σαν shapefile στην περιγραφή του Bot που κάναμε παραπάνω μόνο το DTS χωρίς να χρειαστεί να αναφέρουμε τίποτα άλλο.

## 7.2: Η έννοια του Status

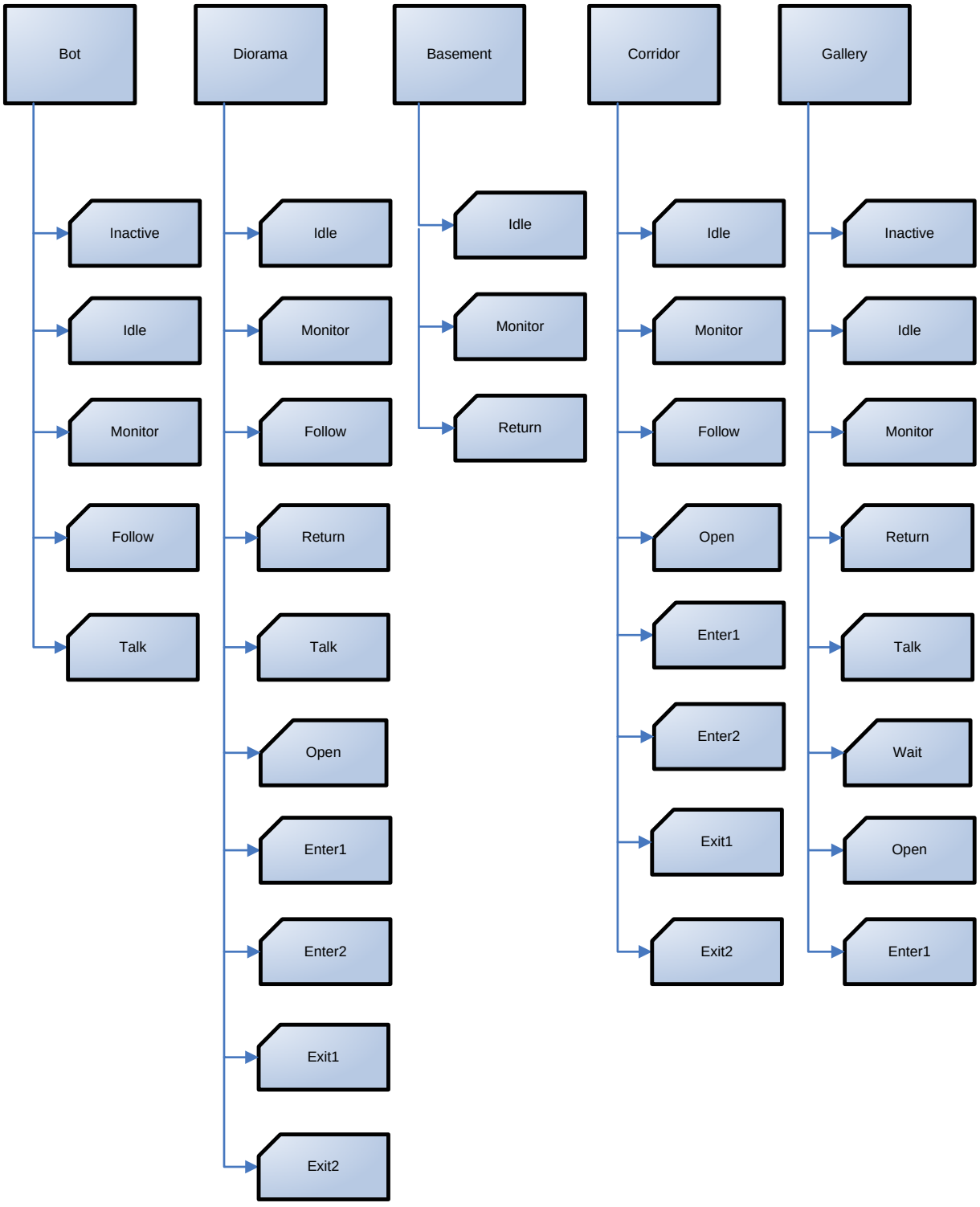
Το status είναι μια από τις μεταβλητές του Bot που έχουν ιδιαίτερη σημασία. Το status είναι ένα string με 2 λέξεις. Η πρώτη λέξη, η οποία μπορεί να απομονωθεί με χρήση της GetWord καθορίζει το δωμάτιο που βρίσκεται το bot. Οι τιμές που λαμβάνει είναι οι:

- **Bot:** αυτή είναι η βασική κατάσταση και με αυτήν ξεκινά το bot στην εφαρμογή. Η κατάσταση αυτή είναι ενεργή όταν το bot βρίσκεται στο δωμάτιο του θόλου.
- **Diorama:** αυτή η κατάσταση ενεργοποιείται όταν ο χρήστης βρίσκεται μέσα στην αίθουσα των εκθεμάτων. Οι λειτουργίες του bot στην αίθουσα εκθεμάτων έχουν πολλές ιδιαιτερότητες, αφού το bot πρέπει να μιλά στον χρήστη όταν αυτός εξετάζει ένα αντικείμενο.
- **Basement:** αυτή η κατάσταση ενεργοποιείται όταν ο χρήστης εισέρχεται στην σκάλα που οδηγεί στο υπόγειο όπου βρίσκεται το γραφείο. Όταν το bot βρίσκεται σε αυτήν την κατάσταση, απλά περιμένει τον χρήστη έξω από την πόρτα της σκάλας.
- **Corridor:** αυτή η κατάσταση ενεργοποιείται όταν ο χρήστης εισέρχεται στον διάδρομο που οδηγεί στην πόρτα εξόδου. Το bot σε αυτήν την περίπτωση απλά ακολουθεί τον χρήστη μέσα στο διάδρομο
- **Gallery:** αυτή η κατάσταση ενεργοποιείται όταν ο χρήστης εισέρχεται στο δωμάτιο με τους πίνακες (και τα γειτονικά δωμάτια μ' αυτό, όπως το δωμάτιο των φωτεινών εφέ). Οι λειτουργίες του bot για αυτήν την κατάσταση είναι περιορισμένες: απλά μιλά στο χρήστη πριν ξεκλειδώσει την πόρτα που οδηγεί στο δωμάτιο με τους πίνακες, και κατόπιν εισέρχεται στο δωμάτιο και περιμένει.

Η 2η λέξη του status εκφράζει την συμπεριφορά του bot. Για παράδειγμα, η λέξη follow φανερώνει ότι το bot αυτή τη στιγμή ακολουθεί τον χρήστη. Ο τρόπος με τον οποίο υλοποιείται μια ενέργεια εξαρτάται και από την 1η λέξη: συνήθως όμως οι συμπεριφορές είναι παρόμοιες ανεξαρτήτως της 1ης λέξης, απλά αλλάζουν οι θέσεις ή οι ατάκες που χρησιμοποιούνται. Έτσι έχουμε τις εξής ενέργειες:

- **Inactive:** Αυτό το status είναι το αρχικό, πριν το Bot ενεργοποιηθεί από τον χρήστη. Υπάρχει μόνο για το Bot status.
- **Idle:** όταν το Bot δεν κάνει τίποτα απολύτως.
- **Monitor:** παρόμοια με την idle, με τη διαφορά ότι το Bot στρέφεται προς την τρέχουσα θέση του χρήστη.
- **Talk:** όταν εμφανίζεται στο HintBox (με χρήση της WriteHint) κάποιες ατάκες που λέει το Bot στον χρήστη. Κάθε ατάκα ακολουθείται από μια παύση κατά τη διάρκεια της οποίας το Bot τίθεται σε monitor. Απ' όλες τις ενέργειες, η talk διαφοροποιείται περισσότερο απ' όλες αναλόγως με την 1η λέξη του status.
- **Follow:** όταν το Bot ακολουθεί τον χρήστη. Η ελάχιστη απόσταση που πρέπει να έχει από το χρήστη ορίζεται σε μια παγκόσμια μεταβλητή.
- **Return:** όταν το Bot κατευθύνεται προς ένα συγκεκριμένο σημείο του δωματίου. Η θέση στην οποία πηγαίνει το Bot εξαρτάται από το δωμάτιο στο οποίο βρισκόμαστε, και άρα στην 1η λέξη του status.
- **Exit1:** αυτή η ενέργεια οδηγεί το Bot σε μια θέση μπροστά στην πόρτα του δωματίου όταν το Bot είναι ήδη μέσα στο δωμάτιο αυτό. Η επόμενη κίνηση του Bot είναι να ανοίξει την πόρτα αν αυτή είναι κλειστή ή να βγει από το δωμάτιο αν είναι ανοιχτή.
- **Exit2:** φυσικό επακόλουθο της exit1, το Bot κατευθύνεται σε μια θέση ακριβώς έξω από το δωμάτιο.
- **Enter1:** όταν το Bot βρίσκεται έξω από ένα δωμάτιο, τότε με την ενέργεια Enter1 κατευθύνεται προς μια θέση μπροστά στην πόρτα έξω από το δωμάτιο. Η επόμενη κίνηση του Bot είναι να ανοίξει την πόρτα αν αυτή είναι κλειστή ή να μπει από το δωμάτιο αν είναι ανοιχτή.

- **Enter2:** φυσικό επακόλουθο της enter1, με αυτήν την ενέργεια το Bot μπαίνει στο δωμάτιο. Η θέση της Enter2 και της Exit1 συμπίπτουν.
- **Open:** η ενέργεια αυτή καλείται μεταξύ της Enter1 και Enter2 και της Exit1 και Exit2 σε περίπτωση που η πόρτα του δωματίου είναι κλειστή. Προφανώς με την ενέργεια αυτή το Bot ανοίγει την πόρτα και μπορεί κατόπιν να μπει ή να βγει.
- **Wait:** αυτή η ενέργεια χρησιμοποιείται μόνο για το gallery status και είναι όμοια με την Monitor. Ο λόγος που εισήχθη είναι γιατί η monitor προϋποθέτει ότι ο χρήστης είναι μέσα στο δωμάτιο, ενώ η wait μπορεί να εφαρμοστεί και εκτός του δωματίου.




**Εικονα 10:** Οι κατηγοριεσ του Bot Status (πρωτη και δευτερη λεξη)

## 7.3: Η έννοια της getNextState

Όταν το Bot ολοκληρώσει τις ενέργειες του status, τότε καλείται η getNextState που αλλάζει το status στην επόμενη κατάσταση. Η αλλαγή αυτή εξαρτάται αφ' ενός από τις 2 λέξεις του status και εφ' ετέρου από άλλους παράγοντες (όπως η θέση του χρήστη ή το state ενός αντικειμένου).

```
function ShowBot::getNextState(%this, %obj, %client){
  %condition = getWord(%obj.status, 0);
  %status = getWord(%obj.status, 1);

  %res_condition = %condition;
  %res_status = %status;

  if( %condition $= "bot" ){
    switch$ (%status) {
      case "inactive":
        %res_status = "talk";
        //%res_condition = "diorama"; // fast DEBUG addition
        //%res_status = "enter1";     // fast DEBUG addition
      case "idle":
        if( %obj.talk_i == 0 ){
          %res_status = "follow";
        } else {
          %res_status = "talk";
        }
      case "monitor":
        if( %obj.talk_i == 0 ){
          %res_status = "follow";
        } else {
          %res_status = "talk";
        }
      case "follow":
        %res_status = "follow";
      case "talk":
        if( %obj.talk_i == 0 ){
          %res_condition = "diorama";
          %res_status = "enter1";
        } else {
          %res_status = "monitor";
        }
      //END OF SWITCH
    }
  }

  if( %condition $= "diorama" ){
    switch$ (%status) {
      case "idle":
        if( %obj.talk_i == 0 ){
          %res_status = "idle";
        } else {
          if( %obj.talk_j == 0 ){
            %res_status = "follow";
          } else {
            %res_status = "talk";
          }
        }
      case "monitor":
        if( %obj.talk_i == 0 ){
          %res_status = "monitor";
        } else {
          if( %obj.talk_j == 0 ){
            %res_status = "follow";
          } else {
            %res_status = "talk";
          }
        }
      case "follow":
```

```
      %res_status = "talk";
    case "return":
      %res_status = "idle";
    case "talk":
      if( %obj.talk_j == 0 ){
        %res_status = "return";
      } else {
        %res_status = "monitor";
      }
    case "exit1":
      if( door_diorama.state $= "open" ){
        %res_status = "exit2";
      } else {
        %res_status = "open";
      }
    case "exit2":
      %res_condition = "bot";
      %res_status = "follow";
    case "open":
      if( door_diorama.state $= "open" ){
        if( %obj.getDatablock().playerInsideDioramaRoom( %client ) ){
          %res_status = "enter2";
        } else {
          %res_status = "exit2";
        }
      }
    case "enter1":
      if( door_diorama.state $= "open" ){
        %res_status = "enter2";
      } else {
        %res_status = "open";
      }
    case "enter2":
      %res_status = "return";
    //END OF SWITCH
  }
}

if( %condition $= "basement" ){
  switch$ (%status) {
    case "idle":
     %res_status = "idle";
    case "monitor":
      %res_status = "monitor";
    case "return":
      %res_status = "idle";
    //END OF SWITCH
  }
}

if( %condition $= "corridor" ){
  switch$ (%status) {
    case "idle":
      %res_status = "idle";
    case "monitor":
      %res_status = "monitor";
    case "follow":
      %res_status = "follow";
    case "return":
      %res_status = "idle";
    case "exit1":
      if( door_corridor.state $= "open" ){
        %res_status = "exit2";
      } else {
        %res_status = "open";
      }
    case "exit2":
      %res_condition = "bot";
      %res_status = "follow";
    case "open":
      if( door_corridor.state $= "open" ){
```

```
        if( %obj.getDatablock().playerInsideCorridorRoom( %client ) ){
          %res_status = "enter2";
        } else {
          %res_status = "exit2";
        }
      }
    case "enter1":
      if( door_corridor.state $= "open" ){
        %res_status = "enter2";
      } else {
        %res_status = "open";
      }
    case "enter2":
      %res_status = "follow";
    //END OF SWITCH
  }
}

if( %condition $= "gallery" ){
  Error("Change State: GALLERY");
  switch$ (%status) {
    case "inactive":
      %res_condition = "bot";
      %res_status = "follow";
    case "idle":
      if( %obj.talk_i == 0 ){
        %res_status = "idle";
      } else {
        if( %obj.talk_j == 0 ){
          Error("Gallery follow (idle)");
          %res_status = "follow";
        } else {
          %res_status = "talk";
        }
      }
    case "monitor":
      if( %obj.talk_i == 0 ){
        %res_status = "idle";
      } else {
        if( %obj.talk_j == 0 ){
          Error("Gallery follow");
          %res_status = "follow";
        } else {
          %res_status = "talk";
        }
      }
    case "return":
      %res_status = "inactive";
    case "talk":
      Error("Change Status: Obj.Talk_j = " @ %obj.talk_j);
      if( %obj.talk_j == 0 ){
        %res_status = "enter1";
      } else {
        %res_status = "wait";
      }
    case "wait":
      %res_status = "talk";
    case "open":
      if( door_corridor.state $= "open" ){
        if( %obj.getDatablock().playerInsideGalleryRoom( %client ) ){
          %res_status = "return";
        } else {
          %res_condition = "bot";
          %res_status = "follow";
        }
      }
    case "enter1":
      if( door_gallery.state $= "open" ){
        %res_status = "return";
      } else {
        %res_status = "open";
```

```
        }
      //END OF SWITCH
    }
  }

  %result= %res_condition SPC %res_status;
  Echo("Status: " @ %obj.status @ " has changed to " @ %result );
  return %result;
}
```

Σε μια γρήγορη επισκόπηση του κώδικα, παρατηρούμε τις εξής αλλαγές στο status:

- **Bot**
  - **Inactive:** Αυτό το status είναι το αρχικό, πριν το Bot ενεργοποιηθεί από τον χρήστη. Το επόμενο status είναι το "bot talk" δηλαδή στην πρώτη ενεργοποίηση το Bot αρχίζει να μιλάει
  - **Idle:** Αλλάζει σε "bot talk" αν το Bot βρίσκεται στη μέση ενός μονολόγου, και σε "bot follow" σε άλλη περίπτωση.
  - **Monitor:** Όμοια με την "bot idle".
  - **Follow:** Αλλάζει σε "bot follow" δηλαδή συνεχίζει να ακολουθεί τον χρήστη.
  - **Talk:** Αλλάζει σε "bot monitor" όταν βρίσκεται στη μέση ενός μονολόγου, ενώ σε άλλη περίπτωση αλλάζει σε "diorama open1" δηλαδή ανοίγει την πόρτα του δωματίου με τα εκθέματα. Ο λόγος για αυτήν την κίνηση θα είναι εμφανής παρακάτω.
- **Diorama**
  - **Idle:** Αλλάζει σε "diorama talk" αν το Bot βρίσκεται στη μέση ενός μονολόγου, σε "diorama follow" αν ο μονόλογος έχει τελειώσει αλλά ο χρήστης εξετάζει ένα αντικείμενο (η πληροφορία αυτή βρίσκεται στην μεταβλητή talk_j). Τέλος σε άλλη περίπτωση το status παραμένει "diorama idle".
  - **Monitor:** Όμοια με το "diorama idle".
  - **Follow:** Όταν το Bot φτάσει στον χρήστη το status αλλάζει σε "diorama talk".
  - **Return:** Όταν το Bot φτάσει στον προορισμό του το status αλλάζει σε "diorama idle".
  - **Talk:** Αν ο μονόλογος έχει τελειώσει το status γίνεται "diorama return", αν όχι γίνεται "diorama monitor". Όπως έχουμε αναφέρει κατά τη διάρκεια του μονολόγου υπάρχουν αλλεπάλληλες αλλαγές μεταξύ του status talk και monitor.
  - **Exit1:** Όταν το Bot φτάσει μπροστά στην πόρτα, αν η πόρτα του δωματίου των εκθεμάτων είναι ανοιχτή τότε το status γίνεται "diorama exit2" αλλιώς το Bot πρέπει να ανοίξει την πόρτα με "diorama open".
  - **Exit2:** Όταν το Bot βγει από το δωμάτιο το status αλλάζει σε "bot follow".
  - **Open:** Όταν ανοίξει η πόρτα, το Bot ελέγχει την θέση του avatar. Αν το avatar βρίσκεται έξω από το δωμάτιο εκθεμάτων το status αλλάζει σε "diorama exit2" αλλιώς αλλάζει σε "diorama enter2". Δηλαδή αν ο χρήστης είναι έξω από το δωμάτιο το Bot βγαίνει και αυτό ενώ αν είναι μέσα το Bot μπαίνει και αυτό.
  - **Enter1:** Όταν το Bot φτάσει μπροστά στην πόρτα, αν η πόρτα του δωματίου των εκθεμάτων είναι ανοιχτή τότε το status γίνεται "diorama enter2" αλλιώς το Bot πρέπει να ανοίξει την πόρτα με "diorama open".
  - **Enter2:** Όταν το Bot μπει στο δωμάτιο το status αλλάζει σε "diorama return" και το bot κατευθύνεται στο βάθος του δωματίου απ' όπου επισκοπεί τις κινήσεις του χρήστη.
- **Basement**
  - **Idle:** Αυτό το status δεν αλλάζει.
  - **Monitor:** Αυτό το status δεν αλλάζει.

- **Return:** Όταν το Bot φτάσει στο σημείο όπου θα περιμένει τον χρήστη να βγει από το δωμάτιο, το status αλλάζει σε “basement idle”.

- **Corridor**
  - **Idle:** Αυτό το status δεν αλλάζει.
  - **Monitor:** Αυτό το status δεν αλλάζει.
  - **Follow:** Αυτό το status δεν αλλάζει (το Bot συνεχίζει να ακολουθεί τον χρήστη).
  - **Exit1:** Όταν το Bot φτάσει μπροστά στην πόρτα, αν η πόρτα του διαδρόμου είναι ανοιχτή τότε το status γίνεται “corridor exit2” αλλιώς το Bot πρέπει να ανοίξει την πόρτα με “corridor open”.
  - **Exit2:** Όταν το Bot βγει από τον διάδρομο το status αλλάζει σε “bot follow”.
  - **Open:** Όταν ανοίξει η πόρτα, το Bot ελέγχει την θέση του avatar. Αν το avatar βρίσκεται έξω από τον διάδρομο το status αλλάζει σε “corridor exit2” αλλιώς αλλάζει σε “corridor enter2”.
  - **Enter1:** Όταν το Bot φτάσει μπροστά στην πόρτα του διαδρόμου, αν η πόρτα αυτή είναι ανοιχτή τότε το status γίνεται “corridor enter2” αλλιώς το Bot πρέπει να ανοίξει την πόρτα με “corridor open”.
  - **Enter2:** Όταν το Bot μπει στο διάδρομο το status αλλάζει σε “corridor follow” και το Bot συνεχίζει να ακολουθεί τον χρήστη.

- **Gallery**
  - **Idle:** Αλλάζει σε “gallery talk” αν το Bot βρίσκεται στη μέση ενός μονολόγου και σε “gallery idle” αν ο μονόλογος έχει τελειώσει.
  - **Monitor:** Όμοια με το “gallery idle”.
  - **Follow:** Αυτό το status δεν αλλάζει (το Bot συνεχίζει να ακολουθεί τον χρήστη).
  - **Return:** Όταν το Bot φτάσει στον προορισμό του το status αλλάζει σε “gallery idle”.
  - **Talk:** Αν ο μονόλογος έχει τελειώσει το status γίνεται “gallery enter1”, αν όχι γίνεται “gallery wait”. Επειδή ο μονόλογος λαμβάνει χώρα έξω από το δωμάτιο με τους πίνακες, αναγκαζόμαστε να εισάγουμε το νέο status wait αντί για το monitor.
  - **Wait:** Επειδή το status γίνεται “gallery wait” μόνο όταν μιλά το Bot το επόμενο status είναι πάντα “gallery talk”.
  - **Open:** Όταν ανοίξει η πόρτα, το Bot ελέγχει την θέση του avatar. Αν ο avatar βρίσκεται έξω από το δωμάτιο πινάκων το status αλλάζει σε “bot follow” αλλιώς αλλάζει σε “gallery return”.
  - **Enter1:** Όταν το Bot φτάσει μπροστά στην πόρτα, αν η πόρτα του δωματίου των πινάκων είναι ανοιχτή τότε το status γίνεται “gallery return” αλλιώς το Bot πρέπει να ανοίξει την πόρτα με “gallery open”.

## 7.4: Έλεγχος θέσης του avatar

Η τεχνητή νοημοσύνη της εφαρμογής έχει 2 ουσιαστικούς ελέγχους για την θέση του avatar: ο πρώτος έλεγχος είναι ουσιαστικά ως προς το πόσο κοντά βρίσκεται ο avatar στο Bot, και χρησιμεύει για να αλλάζει το idle status σε monitor status (δηλαδή να στρέφεται το Bot προς την κατεύθυνση του χρήστη) ενώ ο δεύτερος εξετάζει σε ποιο δωμάτιο βρίσκεται ο χρήστης (καθώς σε κάθε δωμάτιο οι λειτουργίες του Bot αλλάζουν).

Ο πρώτος έλεγχος υλοποιείται ως εξής:

```
  // monitor check: Only when the bot is idle...
  if( %status $= "idle" ){
    %player = %client.player;
    %player_xform = %player.getTransform();
    %sub_vector = VectorSub( %obj.getPosition(), %player_xform );
    %distance = VectorLen( %sub_vector );
    if( %distance < $monitor_maxDistance ){
      Echo("Entering Monitor mode.");
      %obj.status = %condition SPC "monitor";
    }
  }
  // and the opposite...
  if( %status $= "monitor" ){
    %player = %client.player;
    %player_xform = %player.getTransform();
    %sub_vector = VectorSub( %obj.getPosition(), %player_xform );
    %distance = VectorLen( %sub_vector );
    if( %distance > $monitor_maxDistance ){
      Echo("Exiting Monitor mode.");
      %obj.status = %condition SPC "idle";
    }
  }
```

Έτσι, αν η 2η λέξη του status είναι “idle” τότε ελέγχεται η απόσταση μεταξύ του avatar και του Bot. Αν η απόσταση αυτή είναι μικρότερη από μια τιμή που σώζεται στην παγκόσμια μεταβλητή monitor_maxDistance τότε το status αλλάζει σε “monitor”. Αντίθετα, αν το status είναι ήδη “monitor” τότε ελέγχεται και πάλι η απόσταση και αν είναι μεγαλύτερη από την monitor_maxDistance τότε το status αλλάζει σε “idle”.

Ο δεύτερος έλεγχος υλοποιείται ως εξής:

```
function Showbot::playerInsideRoom( %this, %client, %room ){
  %player = %client.player;
  %player_xform = %player.getTransform();
  %pl_x = getWord(%player_xform, 0);
  %pl_y = getWord(%player_xform, 1);
  %pl_z = getWord(%player_xform, 2);

  switch$(%room) {
    case "diorama":
      %bound_min_x = $diorama_bound_min_x;
      %bound_min_y = $diorama_bound_min_y;
      %bound_min_z = $diorama_bound_min_z;
      %bound_max_x = $diorama_bound_max_x;
      %bound_max_y = $diorama_bound_max_y;
      %bound_max_z = $diorama_bound_max_z;
    case "staircase":
      %bound_min_x = $staircase_bound_min_x;
      %bound_min_y = $staircase_bound_min_y;
      %bound_min_z = $staircase_bound_min_z;
      %bound_max_x = $staircase_bound_max_x;
      %bound_max_y = $staircase_bound_max_y;
      %bound_max_z = $staircase_bound_max_z;
    case "corridor":
      %bound_min_x = $corridor_bound_min_x;
```

```
      %bound_min_y = $corridor_bound_min_y;
      %bound_min_z = $corridor_bound_min_z;
      %bound_max_x = $corridor_bound_max_x;
      %bound_max_y = $corridor_bound_max_y;
      %bound_max_z = $corridor_bound_max_z;
   case "gallery":
      %bound_min_x = $gallery_bound_min_x;
      %bound_min_y = $gallery_bound_min_y;
      %bound_min_z = $gallery_bound_min_z;
      %bound_max_x = $gallery_bound_max_x;
      %bound_max_y = $gallery_bound_max_y;
      %bound_max_z = $gallery_bound_max_z;
    //END OF SWITCH
  }

  if( ( %pl_x > %bound_min_x ) && ( %pl_x < %bound_max_x ) &&
      ( %pl_y > %bound_min_y ) && ( %pl_y < %bound_max_y ) &&
      ( %pl_z > %bound_min_z ) && ( %pl_z < %bound_max_z ) ){
    return 1;
  }

  return 0;
}
```

Αναλόγως το όρισμα room, επιλέγονται οι κατάλληλες μεταβλητές για τα όρια του δωματίου στους άξονες X,Y και Z βάσει των αντίστοιχων παγκόσμιων μεταβλητών. Αν η θέση του χρήστη βρίσκεται μέσα στα όρια του νοητού τρισδιάστατου ορθογωνίου που ορίζονται από τα παραπάνω όρια τότε η InsideRoom επιστρέφει 1.

## 7.5: Η συνάρτηση activate

Όπως τα υπόλοιπα αλληλεπιδραστικά αντικείμενα της εφαρμογής, έτσι και το Bot μπορεί να ενεργοποιηθεί από τον χρήστη. Για την ακρίβεια, πριν ενεργοποιηθεί πρώτη φορά από τον χρήστη το Bot είναι inactive, δηλαδή δεν κάνει τίποτα. Όταν ενεργοποιηθεί, καλείται η Think με όρισμα μεταξύ άλλων τον handle του πελάτη που το ενεργοποίησε. Αυτό σημαίνει ότι από εδώ και πέρα το Bot θα μιλάει, θα ακολουθεί και θα ελέγχει μόνο τον πελάτη αυτό.

Όταν το Bot τρέχει ήδη την συνάρτηση think, ενεργοποίηση του καλεί την getNextState σε περίπτωση που το Bot βρίσκεται σε μια από τις «παθητικές» καταστάσεις (idle, monitor, wait). Αυτό είναι ιδιαίτερα χρήσιμο σε μονολόγους του Bot, καθώς ενεργοποίηση του επισπεύδει το τέλος της κατάστασης talk. Ο λόγος που δεν μπορεί ο χρήστης να ενεργοποιήσει το Bot όταν αυτό κινείται (καταστάσεις όπως οι return, follow κλπ.) είναι γιατί πρέπει να ολοκληρώσει την διαδρομή του πριν να μπορέσει να αλλάξει κατάσταση, ειδάλλως τα αποτελέσματα θα είναι απρόβλεπτα.

```
function ShowBot::Activate(%this, %obj, %client){
  %obj.client = %client;              // the bot now "belongs" only to one client. It will
only interact with this client, until a new client activates it.
  %status = getWord(%obj.status, 1);
  if( ( %status $= "inactive" ) || ( %status $= "idle" ) || ( %status $= "monitor" ) || (
%status $= "wait" ) ){
    %obj.status = %this.getNextState( %obj, %client );
    %this.think( %obj, %client );
  }
}
```

## 7.6: Η συνάρτηση think

Στην καρδιά της τεχνητής νοημοσύνης βρίσκεται η συνάρτηση think. Καλείται πρώτη φορά από την activate και κατόπιν καλεί τον εαυτό της ανά 10 δευτερόλεπτα με χρήση της schedule.

Αν το Bot κινείται (δηλαδή το status είναι return ή follow), τότε ελέγχεται η ομαλή κίνησή του. Αν έχει βρει εμπόδιο και δεν μπορεί να κινηθεί (δηλαδή η εντολή checkStuck επιστρέψει 1) τότε αλλάζει κατεύθυνση με την correctDirection. Η correctDirection επιστρέφει 1 σε περίπτωση που το Bot έχει φτάσει στον περιορισμό του, οπότε αλλάζει το status στην επόμενη κατάσταση.

Ο επόμενος έλεγχος είναι έλεγχος δωματίου στο οποίο βρίσκεται ο avatar. Αν ο χρήστης μπει σε ένα δωμάτιο και το status του Bot είναι “bot follow” (δηλαδή πρέπει να ακολουθεί τον χρήστη), τότε το status αλλάζει σε κάποιο που αντιστοιχεί σε status εισόδου στο εν λόγω δωμάτιο. Όμοια, αν το Bot βρίσκεται σε ένα δωμάτιο και ο avatar βγει από αυτό, τότε το status αλλάζει σε αυτό που αντιστοιχεί σε status εξόδου από αυτό το δωμάτιο. Εξαίρεση σ’ αυτό είναι αν το Bot είναι σε κατάσταση μετάβασης (enter1, enter2, open, exit1, exit2), οπότε δεν υπάρχει έλεγχος δωματίου.

Ο επόμενος έλεγχος είναι η εναλλαγή μεταξύ monitor και idle, που εξαρτάται από την απόσταση μεταξύ Bot και avatar. Ο έλεγχος αυτός παρουσιάστηκε στον τμήμα **Έλεγχος θέσης του avatar**.

Μετά από όλη την προετοιμασία αυτή, υπάρχει ο έλεγχος του status μέσα από τον οποίον υλοποιούνται οι ενέργειες ανάλογα με τις 2 λέξεις της μεταβλητής status, όπως αναλύσαμε στο τμήμα **Η έννοια του Status**. Για λόγους οικονομίας χώρου, δε θα αναλύσουμε κάθε status, αλλά μόνο τα πιο ενδιαφέροντα σημεία:

- **Idle:** δεν υπάρχει καθόλου κώδικας για αυτό το status, καθώς το Bot υποτίθεται ότι είναι ανενεργό. Η idle αλλάζει σε monitor με τον έλεγχο θέσης του avatar που αναλύσαμε παραπάνω. Η idle διατηρεί αυτή τη μορφή ανεξαρτήτως της 1ης λέξης του status.
- **Monitor:** καλείται μόνο η βοηθητική συνάρτηση monitor η οποία στρέφει το Bot προς την κατεύθυνση του χρήστη. Η monitor αλλάζει σε idle αν ο avatar απομακρυνθεί πολύ από το Bot. Η monitor διατηρεί αυτή τη μορφή ανεξαρτήτως της 1ης λέξης του status.
- **Talk:** ο μονόλογος του bot βασίζεται σε έναν δισδιάστατο πίνακα από ατάκες. Οι «γραμμές» του πίνακα αυτού αποτελούν ξεχωριστά σύνολα από ατάκες που πρέπει να ειπωθούν με τη σειρά. Για την ανάγνωση του πίνακα αυτού χρησιμοποιούνται οι μεταβλητές talk_i και talk_j (γραμμές και στήλες αντίστοιχα). Κάθε κλήση της talk σημαίνει ότι το Bot πρέπει να πει την επόμενη ατάκα, γι' αυτό και αυξάνεται το talk_j κατά 1. Αν έχουν τελειώσει η ατάκες στην τρέχουσα γραμμή, τότε το Bot δεν μιλάει, μηδενίζονται αμφότερα τα talk_i και talk_j και αλλάζει το status. Αν δεν έχουν τελειώσει οι ατάκες τότε καλείται η writeHint που αναλύσαμε στο σκέλος 4.5 και γράφει την τρέχουσα ατάκα στην οθόνη του χρήστη. Κατόπιν αλλάζει το state σε monitor και για να συνεχίσει να μιλά το Bot προγραμματίζεται με χρήση της schedule να αλλάξει και πάλι το state από monitor σε talk. Στην περίπτωση του "gallery talk" το state αλλάζει σε wait αντί για monitor.
- **Bot_talk:** αυτή είναι μια ειδική περίπτωση της talk που χρειάζεται μια εκτενέστερη εξήγηση. Η εφαρμογή ξεκινά με το Bot να ζητά από τον χρήστη έναν σφραγισμένο φάκελο. Αυτό είναι η 1η γραμμή στον πίνακα bot_quote. Αφού τελειώσει τον μονόλογο αυτόν το talk_i αλλάζει σε 2 και έτσι αν επανενεργοποιηθεί το Bot τότε ξεκινά τη 2η γραμμή από ατάκες, που ενημερώνουν τον χρήστη ότι δεν έχει το κατάλληλο φάκελο. Όταν τελειώσει και με αυτόν τον μονόλογο, το talk_i παραμένει 2, με αποτέλεσμα κάθε προσπάθεια του χρήστη συναντά την ίδια σθεναρή άρνηση από το Bot. Ο μόνος τρόπος να αλλάξει το talk_i σε 3 είναι να ενεργοποιήσει ο χρήστης το Bot τη στιγμή που έχει στο inventory του τον σφραγισμένο φάκελο. Σε αυτήν την περίπτωση το Bot λέει την 3η γραμμή του πίνακα bot_quote δηλαδή συγχαίρει τον χρήστη και τον ενημερώνει ότι η ξενάγηση μπορεί να αρχίσει. Κατόπιν μηδενίζονται τα talk_i, talk_j και αλλάζει το status.
- **Follow:** για να ακολουθήσει το avatar του χρήστη, το Bot αρκεί να βρεί την θέση του avatar και καλεί την βοηθητική συνάρτηση goTo. Αν αυτή επιστρέψει 1 τότε αυτό σημαίνει ότι το Bot έχει πλησιάσει πολύ στον χρήστη και αλλάζει το status. Εως ότου η goTo επιστρέψει 1 το Bot θα κινείται προς τον χρήστη. Η follow διατηρεί αυτή τη μορφή ανεξαρτήτως της 1ης λέξης του status.
- **Return:** όμοια με την follow με τη διαφορά ότι η θέση προς την οποία κατευθύνεται το bot είναι σταθερή και εξαρτάται από το δωμάτιο (δηλαδή την 1η λέξη του status).
- **Enter1, Enter2, Exit1, Exit2:** όμοια με την follow με τη διαφορά ότι η θέση προς την οποία κατευθύνεται το bot είναι σταθερή και εξαρτάται από το δωμάτιο (δηλαδή την 1η λέξη του status). Οι θέσεις αυτές είναι κοντά στην πόρτα, είτε έξω είτε μέσα από το εν λόγω δωμάτιο.
- **Open:** απλά καλείται η activate με όρισμα την πόρτα που πρέπει να ανοίξει το Bot. Η πόρτα αυτή εξαρτάται από το δωμάτιο (δηλαδή την 1η λέξη του status). Το status του Bot αλλάζει μόλις το state της πόρτας γίνει "open" δηλαδή η πόρτα είναι ανοιχτή.
- **Wait:** όμοια με την monitor.

Η συνάρτηση τελειώνει με κλήση της schedule με όρισμα την ίδια την think. Έτσι εξασφαλίζεται ότι σε τακτά χρονικά διαστήματα το Bot θα ελέγχει και θα υλοποιεί τις λειτουργίες του.

```
function ShowBot::think(%this, %obj, %client){
  %condition = getWord(%obj.status, 0);
  %status = getWord(%obj.status, 1);
  //Echo("think: " SPC %condition SPC %status );

  // stuck check: Only when the bot is moving...
  if( ( %status $= "go" ) || ( %status $= "return" ) || ( %status $= "follow" ) ){
    if( %this.checkStuck( %obj ) ){
      if( %this.correctDirection( %obj ) ){
        %this.changeState( %obj, %client );
      }
    }
  }

  // diorama exit check: Only if %condition is "diorama"
  if( %condition $= "diorama" ){
    if( ( %status !$= "exit1" ) && ( %status !$= "exit2" ) && ( %status !$= "open" ) && (
%status !$= "enter1" ) && ( %status !$= "enter2" ) ){
      if( !%obj.getDatablock().playerInsideDioramaRoom( %client ) ){
        %obj.status = "diorama exit1";
      }
    }
  }

  // diorama enter check: Only if %condition is "bot follow"
  if( ( %condition $= "bot" ) && ( %status $= "follow" ) ){
    if( %obj.getDatablock().playerInsideDioramaRoom( %client ) ){
      %obj.status = "diorama enter1";
    }
  }

  // basement exit check: Only if %condition is "basement"
  if( %condition $= "basement" ){
    if( !%obj.getDatablock().playerInsideStaircaseRoom( %client ) ){
      %obj.status = "bot follow";
    }
  }

  // basement enter check: Only if %condition is "bot follow"
  if( ( %condition $= "bot" ) && ( %status $= "follow" ) ){
    if( %obj.getDatablock().playerInsideStaircaseRoom( %client ) ){
      %obj.status = "basement return";
    }
  }

  // corridor exit check: Only if %condition is "corridor"
  if( %condition $= "corridor" ){
    if( ( %status !$= "exit1" ) && ( %status !$= "exit2" ) && ( %status !$= "open" ) && (
%status !$= "enter1" ) && ( %status !$= "enter2" ) ){
      if( !%obj.getDatablock().playerInsideCorridorRoom( %client ) ){
        %obj.status = "corridor exit1";
      }
    }
  }

  // corridor enter check: Only if %condition is "bot follow"
  if( ( %condition $= "bot" ) && ( %status $= "follow" ) ){
    if( %obj.getDatablock().playerInsideCorridorRoom( %client ) ){
      %obj.status = "corridor enter1";
    }
  }

  // gallery exit check: Only if %condition is "gallery"
  if( %condition $= "gallery" ){
    Error("Player left gallery!");
    if( ( %status !$= "exit1" ) && ( %status !$= "exit2" ) && ( %status !$= "open" ) && (
%status !$= "enter1" ) && ( %status !$= "enter2" ) ){
      if( !%obj.getDatablock().playerInsideGalleryRoom( %client ) ){
        // special addition because the "talk" state is done OUTSIDE the gallery room
        if( ( %status !$= "talk" ) && ( %status !$= "wait" ) ){
```

```
          %obj.status = "gallery exit1";
        }
      }
    }
  }

  // gallery enter check: Only if %condition is "bot follow"
  if( ( %condition $= "bot" ) && ( %status $= "follow" ) ){
    if( %obj.getDatablock().playerInsideGalleryRoom( %client ) ){
      Error("Player inside Gallery room!");
      %obj.status = "gallery enter1";
    }
  }

  // monitor check: Only when the bot is idle...
  if( %status $= "idle" ){
    %player = %client.player;
    %player_xform = %player.getTransform();
    %sub_vector = VectorSub( %obj.getPosition(), %player_xform );
    %distance = VectorLen( %sub_vector );
    if( %distance < 10 ){
      Echo("Entering Monitor mode.");
      %obj.status = %condition SPC "monitor";
    }
  }
  // and the opposite...
  if( %status $= "monitor" ){
    %player = %client.player;
    %player_xform = %player.getTransform();
    %sub_vector = VectorSub( %obj.getPosition(), %player_xform );
    %distance = VectorLen( %sub_vector );
    if( %distance > $monitor_maxDistance ){
      Echo("Exiting Monitor mode.");
      %obj.status = %condition SPC "idle";
    }
  }

  if( %condition $= "bot" ){
    switch$ (%status) {
      case "idle":
        // do nothing, apparently
      case "monitor":
        %this.monitor( %obj, %client, 0 );
        //%obj.getDatablock().turnTo( %obj, %target );
      case "talk":
        // this is the only way the bot will proceed to talk_state 3
        if( ( %obj.talk_i == 2 ) && ( %obj.talk_j == 0 ) && ( %client.inventory ) && (
%client.inventory.getDatablock() == DB_envelope_sealed.getID() ) ){
          %client.inventory = 0;
          commandToClient( %client, 'setGUIbitmap', "GunSight",
"control/client/ifaces/pointer.png");
          %obj.talk_i = 3;
        }
        %this.monitor( %obj, %client, 0 );
        %obj.talk_j = %obj.talk_j + 1;
        Echo("Talk (i,j) = " @ %obj.talk_i SPC %obj.talk_j );
        if( $bot_quote[ %obj.talk_i, %obj.talk_j ] $= "" ){
          if( %obj.talk_i == 1 ){
           %obj.talk_i = 2;
           %obj.talk_j = 0;
            %this.changeState( %obj, %client );
          } else if( %obj.talk_i == 2 ){
           %obj.talk_i = 2;   // only one thing can change this talk_status
           %obj.talk_j = 0;
            %this.changeState( %obj, %client );
          } else if( %obj.talk_i == 3 ){
           %obj.talk_i = 0;
           %obj.talk_j = 0;
            %this.changeState( %obj, %client );
          }
        } else {
```

```
          // emotes add-on
          if( %obj.talk_j == 1 ){
            if( %obj.talk_i == 1 ){
              %obj.setActionThread("celwave");
            } else if( %obj.talk_i == 2 ){
              %obj.setActionThread("death5");
            } else if( %obj.talk_i == 3 ){
              %obj.setActionThread("death4");
            }
          }
          commandToClient( %client, 'writeHint', $bot_quote[ %obj.talk_i, %obj.talk_j ]
);
          %this.changeState( %obj, %client );
          %this.schedule(4500, "changeState", %obj, %client);
        }
      case "follow":
        %player = %client.player;
        %target = %player.getTransform();
        //Echo( "The player is at " @ %target );
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Stopped following.");
        }
      // END OF SWITCH
    }
  }
  if( %condition $= "diorama" ){
    switch$ (%status) {
      case "idle":
        // do nothing, apparently
      case "monitor":
        %this.monitor( %obj, %client, 0 );
      case "talk":
        %this.monitor( %obj, %client, 0 );
        %obj.talk_j = %obj.talk_j + 1;
        Echo("Talk (i,j) = " @ %obj.talk_i SPC %obj.talk_j );
        if( $diorama_quote[ %obj.talk_i, %obj.talk_j ] $= "" ){
          %obj.talk_i = 0;
          %obj.talk_j = 0;
          %this.changeState( %obj, %client );
        } else {
          commandToClient( %client, 'writeHint', $diorama_quote[ %obj.talk_i, %obj.talk_j
]);
          %this.changeState( %obj, %client );
          %this.schedule(4500, "changeState", %obj, %client);
        }
      case "follow":
        %player = %client.player;
        %target = %player.getTransform();
        //Echo( "The player is at " @ %target );
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Stopped following.");
        }
      case "return":
       %target = $diorama_monitor;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Stopped returning.");
        }
      case "exit1":
        %target = $diorama_inside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "exit2":
        %target = $diorama_outside;
```

```
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "open":
        // the bot can unlock all doors in the museum
        if( door_diorama.state $= "locked" ){
          door_diorama.state = "closed";
        }
        if( door_diorama.state $= "closed" ){
          %obj.setActionThread("death2");
          door_diorama.getDatablock().activate( door_diorama, %client);
        }
        if( door_diorama.state $= "open" ){
          %this.changeState( %obj, %client );
        }
      case "enter1":
        %target = $diorama_outside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "enter2":
        %target = $diorama_inside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      // END OF SWITCH
    }
  }
  if( %condition $= "basement" ){
    switch$ (%status) {
      case "idle":
        // do nothing, apparently
      case "monitor":
        %this.monitor( %obj, %client, 0 );
      case "return":
       %target = $staircase_monitor;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Started waiting.");
        }
      // END OF SWITCH
    }
  }
  if( %condition $= "corridor" ){
    switch$ (%status) {
      case "idle":
        // do nothing, apparently
      case "monitor":
        %this.monitor( %obj, %client, 0 );
      case "return":
       %target = $corridor_monitor;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Started waiting.");
        }
      case "follow":
        %player = %client.player;
        %target = %player.getTransform();
        //Echo( "The player is at " @ %target );
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Stopped following.");
        }
      case "exit1":
        %target = $corridor_inside;
```

```
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "exit2":
        %target = $corridor_outside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "open":
        // the bot can unlock all doors in the museum
        if( door_corridor.state $= "locked" ){
          door_corridor.state = "closed";
        }
        if( door_corridor.state $= "closed" ){
          %obj.setActionThread("death2");
          door_corridor.getDatablock().activate( door_corridor, %client);
        }
        if( door_corridor.state $= "open" ){
          %this.changeState( %obj, %client );
        }
      case "enter0":
        %target = $corridor_inside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "enter1":
        %target = $corridor_outside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "enter2":
        %target = $corridor_inside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      // END OF SWITCH
    }
  }
  if( %condition $= "gallery" ){
    switch$ (%status) {
      case "idle":
        // do nothing, apparently
      case "monitor":
        %this.monitor( %obj, %client, 0 );
      case "return":
       %target = $gallery_monitor;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Started waiting.");
        }
      case "talk":
        %this.monitor( %obj, %client, 0 );
        %obj.talk_j = %obj.talk_j + 1;
        Echo("Talk (i,j) = " @ %obj.talk_i SPC %obj.talk_j );
        if( $gallery_quote[ %obj.talk_i, %obj.talk_j ] $= "" ){
          %obj.talk_i = 0;
          %obj.talk_j = 0;
          Error("Dialog End");
          %this.changeState( %obj, %client );
        } else {
          commandToClient( %client, 'writeHint', $gallery_quote[ %obj.talk_i, %obj.talk_j
]);
          %this.changeState( %obj, %client );
          %this.schedule(4500, "changeState", %obj, %client);
        }
```

```
      case "wait":
        %this.monitor( %obj, %client, 0 );
      case "follow":
        %player = %client.player;
        %target = %player.getTransform();
        //Echo( "The player is at " @ %target );
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
          Echo("Stopped following.");
        }
      case "exit1":
        %target = $gallery_inside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "exit2":
        %target = $gallery_outside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "open":
        // the bot can unlock all doors in the museum
        if( door_gallery.state $= "locked" ){
          door_gallery.state = "closed";
        }
        if( door_gallery.state $= "closed" ){
          %obj.setActionThread("death2");
          door_gallery.getDatablock().activate( door_gallery, %client);
        }
        if( door_gallery.state $= "open" ){
          %this.changeState( %obj, %client );
        }
      case "enter1":
        %target = $gallery_outside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      case "enter2":
        %target = $gallery_inside;
        %result = %this.goTo( %obj, %target );
        if( %result ){
          %this.changeState( %obj, %client );
        }
      // END OF SWITCH
    }
  }

  %this.schedule($think_interval, "think", %obj, %client);
}
```

## 7.7: Βοηθητικές συναρτήσεις

Για τις ανάγκες της κίνησης του Bot υπάρχουν οι βοηθητικές συναρτήσεις goTo, checkStuck, correctDirection.

Η συνάρτηση goTo δίνει εντολή να κινηθεί το Bot προς ένα σημείο στην εφαρμογή (είτε αυτό το σημείο είναι σταθερό είτε κινούμενο όπως ο χρήστης). Η goTo υπολογίζει το διάνυσμα μεταξύ του Bot και του σημείου-στόχου, και κατασκευάζει ένα διάνυσμα που είναι υποδιαίρεση αυτού. Έτσι, μεταξύ δύο κλήσεων της think το Bot καλείται να κινηθεί μόνο μια μικρή απόσταση προς το σημείο-στόχο. Επίσης σε περίπτωση που το Bot φτάσει πολύ κοντά στο στόχο του (η ελάχιστη απόσταση ορίζεται στις παγκόσμιες μεταβλητές $follow_minDistance αν ακολουθεί τον avatar και $goTo_minDistance αν το σημείο είναι στατικό) τότε η goTo επιστρέφει 1 επιτρέποντας στην think συνάρτηση να αντιληφθεί ότι το Bot έφτασε και να αλλάξει το status.

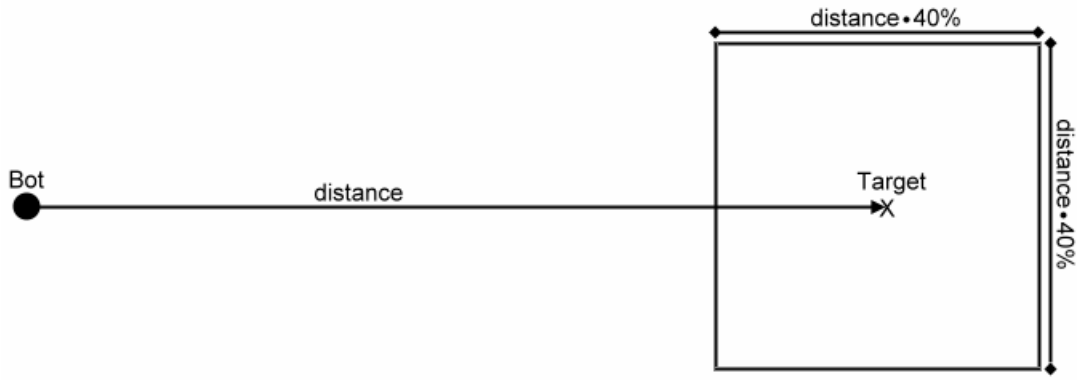
```
function ShowBot::goTo(%this, %obj, %target){
  //Echo("Target = " @ %target );
  %obj.target = %target;

  // determine the distance between target and object
  %sub_vector = VectorSub( %obj.getPosition(), %target );
  %distance = VectorLen( %sub_vector );
  //Echo( "Distance between " @ %obj @ " and (" @ %target @ ") is " @ %distance @ " and
the sub_vector is " @ %sub_vector );

  if( getWord(%obj.status, 1) $= "follow" ){
    %min_distance = $follow_minDistance;
  } else {
    %min_distance = $goTo_minDistance;
  }

  if( %distance < %min_distance ){
    // reached destination: inform the calling function
    Echo("Destination reached");
    return 1;
  } else {
    // continue moving
    %move_step = VectorScale( %sub_vector, 0.5 );
    while( VectorLen( %move_step ) > 5 ){
      %move_step = VectorScale( %move_step, 0.5 );
    }
  }
  %temp_target = VectorSub( %obj.getPosition(), %move_step );
  %obj.temp_target = %temp_target;
  Echo( "The move_step is: " @ %move_step @ " and the temp_target is: " @ %temp_target );
  %obj.setMoveDestination( %temp_target );
  return 0;
}
```

Η συνάρτηση checkStuck ελέγχει αν το Bot έχει κάνει μια ελάχιστη απόσταση από την προηγούμενη φορά που κλήθηκε η checkStuck (η checkStuck καλείται μέσα στην think). Ελέγχονται λοιπόν οι αποστάσεις μεταξύ των σημείων που σώζονται στις βοηθητικές μεταβλητές last_position και current_position. Αν η απόσταση τους είναι μικρότερη από την τιμή της παγκόσμιας μεταβλητής stuck_maxDistance τότε επιστρέφει 1 στην think οπότε πρέπει να κληθεί η correctDirection για να γίνουν προσπάθειες να ξεκολλήσει από το σημείο που έχει σταματήσει.

```
function ShowBot::checkStuck(%this, %obj ){
  %obj.last_position = %obj.current_position;
  %obj.current_position = %obj.getPosition();

  %obj_pos = %obj.getPosition();
  %obj_x = getWord(%obj_pos, 0);
```

```
  %obj_y = getWord(%obj_pos, 1);
  %obj_z = getWord(%obj_pos, 2);

  %last_pos = %obj.last_position;
  %tar_x = getWord(%last_pos, 0);
  %tar_y = getWord(%last_pos, 1);
  %tar_z = getWord(%last_pos, 2);

  %sub_vector = VectorSub( %obj_pos, %last_pos );
  %distance = VectorLen( %sub_vector );
  //Echo( "Distance between current position and last position is " @ %distance @ " and
the sub_vector is " @ %sub_vector );

  if( %distance < $stuck_maxDistance ){
    Echo( "Bot Stuck." );
    return 1;
  }
  return 0;
}
```

Η συνάρτηση correctDirection αλλάζει τον στόχο προς τον οποίο κινείται το Bot, και τον τοποθετεί σε κάποιο άλλο σημείο που επιλέγεται σχεδόν τυχαία. Αφού ελεγχθεί ότι το Bot δεν έχει φτάσει ακόμα στον προορισμό του, ο στόχος μετακινείται κατά -20% έως 20% της απόστασης του από το Bot στο επίπεδο XY και δίνεται εντολή στο Bot να κινηθεί προς τον νέο στόχο. Αν ο νέος στόχος επιλέχθηκε εσφαλμένα, θα ξαναεπιστρέψει η checkStuck 1 στην επόμενη κλήση της think οπότε θα κληθεί η correctDirection και πάλι για να δοκιμάσει νέο στόχο.




**ΕΙΚΟΝΑ11:** Η ΛΕΙΤΟΥΡΓΙΑ ΤΗΣ CORRECTDIRECTION

```
function ShowBot::correctDirection(%this, %obj ){
  // We currently do not need to search for an alternative Z-position

  %x = getWord(%obj.target, 0);
  %y = getWord(%obj.target, 1);
  %z = getWord(%obj.target, 2);

  %bot_pos = %obj.getPosition();
  %bot_x = getWord(bot_pos, 0);
  %bot_y = getWord(bot_pos, 1);
  //%bot_z = getWord(bot_pos, 2);

  %sub_vector = VectorSub( %bot_pos, %obj.target );
  %distance = VectorLen( %sub_vector );
  //Echo("Distance between " @ %sub_vector @ " and " @ %obj.target @ " is " @ %distance @
" (" @ %sub_vector @ ").");

  if( getWord(%obj.status, 1) $= "follow" ){
    %min_distance = $follow_minDistance;
  } else {
    %min_distance = $goTo_minDistance;
  }

  if( %distance < %min_distance ){
    Echo( %obj.getDataBlock().getName() @ ", time to stop!" );
    return 1;
    //%obj.setMoveDestination( %bot_pos );
  } else {
    %rnd1 = ( getrandom(40) - 20 ) / 100;
    %rnd2 = ( getrandom(40) - 20 ) / 100;
    //%rnd3 = getrandom(20) - 10;

    %new_x = %x + %rnd1 * %distance;
    %new_y = %y + %rnd2 * %distance;
    //%new_z = %z*( 100 + %rnd3 )/100;

    %new_direction = %new_x SPC %new_y SPC %z;
    //Echo( %obj.getDataBlock().getName() @ " will now move to " @ %new_direction );
    %obj.setMoveDestination( %new_direction );
  }
}
```

Η συνάρτηση monitor καλείται όταν το Bot έχει ως 2η λέξη του status του τη λέξη "monitor" ή "talk". Επειδή το Bot πρέπει να κοιτά τον χρήστη όταν του μιλά ή όταν τον παρατηρεί, η monitor καλείται ανά μικρότερα διαστήματα από την think. Έτσι, καλείται κάθε 0.5sec και εξακολουθεί να καλεί αναδρομικά τον εαυτό της έως ότου έρθει η στιγμή να κληθεί η think. Αν το status είναι ακόμα monitor, τότε η think θα ξανακαλέσει την monitor που θα τρέχει μέχρι την επόμενη κλήση της think κ.ο.κ.

```
function ShowBot::monitor(%this, %obj, %client, %time){
  if( ( getWord(%obj.status, 1) $= "monitor" ) || ( getWord(%obj.status, 1) $= "talk" )
){
    if( %time < $think_interval ){
      %player = %client.player;
      %this.turnTo(%obj, %player.getTransform() );
      %time = %time + 50;
      %this.schedule(50, "monitor", %obj, %client, %time);
    }
  }
}
```

Η συνάρτηση turnTo θέτει το rotation του Bot έτσι ώστε να είναι στραμμένο προς τη θέση του avatar. Με χρήση της τριγωνομετρικής συνάρτησης mAtan αυτό επιτυγχάνεται με ελάχιστες γραμμές κώδικα.

```
function ShowBot::turnTo(%this, %obj, %target){
  %obj_xform = %obj.getTransform();
  %obj_x = GetWords(%obj_xform,0);
  %obj_y = GetWords(%obj_xform,1);

  %target_x = GetWords(%target,0);
  %target_y = GetWords(%target,1);

  //Echo( %obj.getID() @ " looks at (" @ %target_x @ ", " @ %target_y @ ")");
  %d_x = %target_x - %obj_x;
  %d_y = %target_y - %obj_y;

  %turn = mAtan(%d_x, %d_y);
  %new_obj_xform = GetWords(%obj_xform,0,2) SPC "0 0 1" SPC %turn;
  //Echo( %obj.getID() @ " turns by " @ %turn @ ". (d_x d_y) = (" @ %d_x SPC %d_y@ ")");
  //Echo( %obj @ " turns by " @ %turn);
  //Echo( "New Xform is " @ %new_obj_xform);
  %obj.setTransform(%new_obj_xform);
}
```

# Βιβλιογραφία - Ιστοσελίδες


- **3D Game Programming all in one** (Kenneth Finney)
- **Advanced 3D Game Programming all in one** (Kenneth Finney)
- **GarageGames** (http://www.garagegames.com)
- **Hall of Worlds** (http://www.hallofworlds.com)
- **Games Blog** (http://www.jamestadeo.com/blog_games/)
- **Torque Tutorials** (http://holodeck.st.usm.edu/vrcomputing/vrc_t/tutorials/)
- **Minions of Mirth Game** (http://www.prairiegames.com/)
- **Wikipedia: Torque Game Engine** (http://en.wikipedia.org/wiki/Torque_Game_Engine)