**επεξεργασια φυσικησ γλωσσασ**

Εργασία σε C# με αναγνώριση φωνής και Text-To-Speech

ΕΙΣΑΓΩΓΗ

# Σκοπός της Εφαρμογής

Η εφαρμογή έχει ως σκοπό την επίδειξη της επεξεργασίας φωνής και φυσικής γλώσσας, χρησιμοποιώντας την C#. **Κύριος σκοπός ήταν να φανεί η ομιλία από τον χρήστη προς τον υπολογιστή και η ανταπόκριση του υπολογιστή προς τον χρήστη.** Η συγκεκριμένη γλώσσα μάς παρέχει με τις κατάλληλες λειτουργίες, ώστε να αναγνωρίζεται η φωνή και, παράλληλα, να αναπαράγεται μία «ρομποτική φωνή» ως ανταπόκριση. Για να επιτύχουμε την επίδειξη, χρησιμοποιείται το βιντεοπαιχνίδι Keep Talking and Nobody Explodes, το οποίο ταιριάζει βέλτιστα για τον σκοπό αυτό.

# Γιατί Επελέγη το Συγκεκριμένο Βιντεοπαιχνίδι;

Σε πρώτη φάση, θα πρέπει να καταλάβουμε πώς παίζεται το συγκεκριμένο παιχνίδι. Το παιχνίδι, συνήθως, παίζεται με δύο (ή περισσότερους) παίκτες. Ο ένας από αυτούς έχει στα χέρια του μια βόμβα, την οποία πρέπει να αφοπλίσει. Οι υπόλοιποι παίκτες, κρατούν ένα εγχειρίδιο, το οποίο περιέχει τις οδηγίες, ώστε να αφοπλιστεί η βόμβα. **Κανένας από τους δύο παίκτες δεν έχει οπτική επαφή με τον άλλον. Στόχος είναι να αφοπλιστεί η βόμβα, χωρίς να κοιτάξει κανείς τα εργαλεία του άλλου.** Για παράδειγμα, δεν μπορεί αυτός που αφοπλίζει την βόμβα να κοιτάξει το εγχειρίδιο. Ούτε αυτός που έχει το εγχειρίδιο επιτρέπεται να κοιτάξει την ίδια την βόμβα. **Οπότε, η ουσιαστική χρήση της εφαρμογής αυτής, είναι η αυτοματοποίηση του εγχειριδίου μέσω ενός BOT. Θα μιλάει ο χρήστης στο BOT και αυτό θα του δίνει τις κατάλληλες οδηγίες για την αφόπλιση της βόμβας.**

# Υλοποίηση

Χρησιμοποιώντας το .NET Framework, φτιάχτηκε μια εφαρμογή σε Windows Forms, στην οποία ο χρήστης μπορεί να μιλάει και να παίρνει απαντήσεις ανάλογα με το τι λέει. Η εφαρμογή είναι φτιαγμένη για να τρέχει σε Windows (κατά προτίμηση από Windows 7 και έπειτα) και χρησιμοποιεί το προεπιλεγμένο μικρόφωνο του χρήστη, για να παίρνει τις εντολές. Το back-end της εφαρμογής είναι φτιαγμένο σε C#, χρησιμοποιώντας κυρίως τις κανονικές κλάσεις της γλώσσας για την αναγνώριση φωνής.

ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΗΝ ΕΦΑΡΜΟΓΗ

Για να χρησιμοποιηθεί ορθώς η εφαρμογή, θα πρέπει να ακολουθηθούν επ’ ακριβώς οι φωνητικές εντολές, ώστε να πάρει τα σωστά αποτελέσματα ο χρήστης. Επίσης, ο χρήστης συνίσταται να ‘χει μια βασική ιδέα στο πώς λύνονται τα modules, προτού δοκιμάσει να τα λύσει. Πατώντας εδώ, ο χρήστης μπορεί να δει ακριβώς τις φωνητικές εντολές, αλλά και τις ενέργειες στις οποίες χρειάζεται να προβεί, ώστε να πάρει τις σωστές απαντήσεις από το BOT.

# Πρώτη Οθόνη

Ένα από τα πρώτα πράγματα που θα αντικρίσει ο χρήστης, ανοίγοντας, την εφαρμογή, είναι δύο πεδία. Δεν μπορεί να γράψει σε κανένα από τα δύο, παραμόνο να βλέπει τις εντολές που λέει και τις εντολές που ακούει από το BOT.

Graphical user interface, text, application

Description automatically generated

Η ΜΕΤΑΤΡΟΠΗ ΣΕ ΚΩΔΙΚΑ

Για να επιτύχουμε την έκδοση σωστών απαντήσεων από το BOT, χρειάζεται προτίστως να «μεταφράσουμε» το εγχειρίδιο της βόμβας σε κώδικα. **Για να γίνει αυτό, χρησιμοποιήθηκαν οι αντικειμενοστρεφείς τεχνικές της C#.** Στο παιχνίδι, η βόμβα έχει μερικά τμήματα (modules), που αφοπλίζονται. Όταν αφοπλιστούν όλα τα modules, αφοπλίζεται ολόκληρη η βόμβα. Άρα, **γενικεύοντας την έννοια του module και την έννοια της βόμβας, μπορούμε να απεικονίσουμε σε κώδικα το εγχειρίδιο της βόμβας.** Επίσης, η βόμβα έχει κάποιες καθοριστικές ιδιότητες. Όπως, π.χ., ο αριθμός μπαταριών που έχει, το τελευταίο ψηφίο του σειριακού της αριθμού κτλ. Αυτές οι ιδιότητες είναι σημαντικές πληροφορίες για κάποια modules, καθώς καθορίζεται η λύση τους από αυτές.

# Απεικόνιση της Βόμβας

Όπως αναφέρθηκε και πριν, η βόμβα έχει κάποιες σημαντικές ιδιότητες. Αυτές είναι οι εξής:

* Ο ακριβής αριθμός μπαταριών
* Το τελευταίο ψηφίο του σειριακού αριθμού (αν είναι περιττό ή άρτιο)
* Η παρουσία μιας φωτεινής ετικέτας που γράφει «FRK»
* Η παρουσία μιας φωτεινής ετικέτας που γράφει «CAR»
* Η παρουσία μιας οποιασδήποτε παράλληλης θύρας
* Η παρουσία οποιουδήποτε φωνήεντος στο σειριακό

**Οπότε, η κλάση της βόμβας θα έχει τον εξής κώδικα:**

public class Bomb  
{  
 //bomb properties  
 public readonly int Batteries;  
 public readonly bool Parallel;  
 public readonly bool FRK;  
 public readonly bool CAR;  
 public readonly bool Vowel;  
 public readonly bool EvenDigit;  
  
 public Bomb(int batteries, bool parallel, bool frk, bool car, bool vowel, bool evenDigit)  
 {  
 Batteries = batteries;  
 Vowel = vowel;  
 Parallel = parallel;  
 EvenDigit = evenDigit;  
 FRK = frk;  
 CAR = car;  
 }  
}

Όλα τα μέλη της κλάσης Bomb είναι καθοριστικά για το πως φτάνουν στη λύση τους ορισμένα modules. Επίσης, τα μέλη είναι αδιάλλακτα και παίρνουν τιμή με την συνάρητηση κατασκευαστή (constructor). Θα δούμε και παρακάτω πώς επιδρούν και οι υπόλοιπες πληροφορίες της βόμβας τα modules.

# Γενίκευση του Module

Έχοντας υπ’όψιν ότι οι ιδιότητες της Βόμβας επηρεάζουν το πώς επιλύεται ένα module, φτιάχνουμε μια abstract κλάση KTANE\_Module, **η οποία έχει σαν πεδίο ένα αντικείμενο Βόμβας μέσα της.** Έτσι μπορούμε να παρέχουμε τις πληροφορίες της βόμβας στα modules κατ’ ευθείαν από την ίδια την abstract κλάση.

public abstract class KTANE\_Module  
{  
 private protected Bomb \_bomb;  
  
 private protected KTANE\_Module(Bomb bomb)  
 {  
 \_bomb = bomb;  
 }  
  
 public abstract string Solve();  
}

Όπως φαίνεται και παραπάνω, κάθε φορά που φτιάχνουμε ένα module, **πρέπει οπωσδήποτε να φτιαχτεί και το στιγμιότυπο της βόμβας, ώστε να μπορούμε να παίρνουμε τις πληροφορίες της.** Επίσης, αυτή η κλάση έχει και μία abstract συνάρτηση, την Solve, η οποία επιστρέφει τις οδηγίες ώστε να επιλυθεί το Module.

# Υλοποιήσεις Όλων των Modules

Χρησιμοποιώντας την παραπάνω γενίκευση του KTANE\_Module, έγιναν 12 διαφορετικές κλάσεις για 12 modules. Όλες αυτές οι κλάσεις μπορούν