ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη λογισμικού υπολογισμού γεωμετρικών αναπτυγμάτων με τη βοήθεια τρισδιάστατης μοντελοποίησης

ΜΑΡΑΓΚΟΥΔΑΚΗΣ ΑΝΤΩΝΙΟΣ - ΕΜΜΑΝΟΥΗΛ

Εξεταστική Επιτροπή

ΚΑΘΗΓΗΤΗΣ ΖΕΡΒΑΚΗΣ ΜΙΧΑΛΗΣ (ΗΜΜΥ)

ΟΜΟΤΙΜΟΣ ΚΑΘΗΓΗΤΗΣ ΚΑΛΑΪΤΖΑΚΗΣ ΚΩΣΤΑΣ (ΗΜΜΥ)

ΚΑΘΗΓΗΤΗΣ ΑΝΤΩΝΙΑΔΗΣ ΑΡΙΣΤΟΜΕΝΗΣ (ΜΠΔ)

Χανιά, Δεκέμβριος 2022

TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Software development for the calculation of geometric development via three-dimensional modeling

MARAGKOUDAKIS ANTONIOS - EMMANOUIL

Thesis Committee

PROFESSOR ZERVAKIS MICHALIS (ECE)

PROFESSOR EMERITUS KALAITZAKIS KOSTAS (ECE)

PROFESSOR ANTONIADIS ARISTOMENIS (PEM)

Chania, December 2022

Περίληψη

Στην επιστήμη των υπολογιστών μια διεπαφή προγραμματισμού εφαρμογών (Application Programming Interface - API), επιτρέπει σε μία εφαρμογή να εξάγει πληροφορίες από ένα κομμάτι λογισμικού και να τις χρησιμοποιεί σε μια άλλη εφαρμογή. Συχνά σκοπός δεν είναι μόνο η εξαγωγή των πληροφοριών αυτών αλλά και η εκτέλεση ενεργειών όπως η κλήση ρουτίνων σε μία εφαρμογή.

Στην παρούσα διπλωματική υλοποιήθηκε ένα API το οποία είχε ως σκοπό να δέχεται είσοδο από έναν χρήστη και να καλεί ρουτίνες εντός της εφαρμογής τρισδιάστατης μοντελοποίησης Autodesk Inventor. Δημιουργήθηκε δηλαδή ένα Inventor API, το οποίο συντελείται από μεθόδους που υλοποιήθηκαν και καλούνται – για πέντε περιπτώσεις Κυλινδρικών Τομών - με σκοπό την κατασκευή των επιφανειακών τρισδιάστατων μοντέλων, την ανάπτυξη αυτών στον δισδιάστατο χώρο και εν συνέχεια τον υπολογισμό των δισδιάστατων αναπτυγμάτων. Τέλος, επιπλέον μέθοδοι υλοποιήθηκαν και καλούνται ως επακόλουθο των προαναφερθέντων, με καταληκτικό σκοπό τους, την δημιουργία του Μηχανολογικού Σχεδίου των αναπτυγμάτων των Κυλινδρικών Τομών.

Η συνολική εργασία λοιπόν, έχει σαν σκοπό την ανάπτυξη της προπεριγραφείσας εφαρμογής , η οποία και υλοποιήθηκε εντός του Microsoft Visual Studio, το οποίο είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (Integrated Development Environment - IDE). Στο Microsoft Visual Studio δύο ήταν τα βασικά μέρη που αναπτύχθηκαν. Μία γραφική διεπαφή χρήστη και οι ρουτίνες όπου παράγουν τα επιθυμητά αντικείμενα εντός του Autodesk Inventor. Για την ανάπτυξη της γραφικής διεπαφής χρησιμοποιήθηκε η πλατφόρμα των Windows Forms, ενώ για τον προγραμματισμό των ενεργειών του Inventor API χρησιμοποιήθηκε η βιβλιοθήκη autodesk.inventor.interop.dll, η οποία είναι μία βιβλιοθήκη δυναμικής σύνδεσης (Dynamic Link Library - DLL). Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι η Visual Basic και επομένως η εφαρμογή αυτή μπορεί να θεωρηθεί στο σύνολό της ένα Inventor VBA (Visual Basic Application).

Η γραφική διεπαφή χρήστη που δημιουργήθηκε περιγράφεται από το αγγλικό ακρωνύμιο GUI (Graphical User Interface). Ένα GUI ενσωματώνεται σε μια εφαρμογή έτσι ώστε να της επιτρέπει να περιέχει γραφικά στοιχεία όπως εικονίδια, μενού, πλαίσια ελέγχου και άλλα τα οποία την καθιστούν ευκολότερη στην αλληλεπίδραση του χρήστη. Στην προκειμένη περίπτωση το GUI μπορεί να θεωρηθεί ως μία διεπαφή - μάσκα η οποία είναι απλή για τον χρήστη, ενώ παράλληλά έχει σαν σκοπό να κρύψει από αυτόν στο background ότι πραγματικά εκτελείτε εντός του Inventor και μόνο να του παραθέτει τα αποτελέσματα προς αποθήκευση.

Στην παρούσα διπλωματική θα αναλυθούν ο τρόπος με τον οποίο το GUI συνεργάζεται με το API καθώς και το πως το API καλεί μία σειρά από ρουτίνες εντός του Inventor. Θα αναλυθεί ο σκοπός της κάθε ρουτίνας που δημιουργήθηκε, δηλαδή, το τι αυτή δημιουργεί εντός του Inventor, για ποιόν λόγο το δημιουργεί ή τι προβλήματα λύνει. Τέλος θα ακολουθήσουν τα συμπεράσματα καθώς και θα γίνει λόγος για πιθανές μελλοντικές εργασίες που είναι δυνατόν να επακολουθήσουν της εργασίας αυτής.

Abstract

In computer science, an Application Programming Interface (API) allows an application to extract information from one piece of software and use it in another. Often the purpose is not only to extract this information but also to perform actions such as calling routines within an application.

In this thesis an Inventor API which calls routines within the 3D modeling application Autodesk Inventor was created. During operation, routines are called, which generate Mechanical Drawing of the 2D development for five cases of cylindrical intersection. In short, the API builds a 3D model based on the user input, develops it in 2D space and outputs intersection specifications as well as the original 3D model within the mechanical drawing generated.

Overall, the principal aim of this thesis was to develop the aforementioned application, this was implemented in two main parts using the Integrated Development Environment (IDE), Microsoft Visual Studio. The first of these was, a user interface. The second was the routines that actually produce the desired objects within Inventor. The interface was developed using the Windows Forms platform, while the Inventor API was developed using the Dynamic Link Library (DLL) autodesk.inventor.interop.dll. The language used to develop the application is Visual Basic and therefore this application can be considered as an Inventor VBA (Visual Basic Application).

The interface created is essentially a Graphical User Interface (GUI). A GUI is defined as an interface which facilitates the user through the use of graphical elements such as icons, menus, check boxes, and others. In this thesis the GUI acts as a user-friendly mask. It does this by hiding the actual process executed in Inventor from the user and only presenting him with the finished product of the calculation and modeling.

This thesis will expand upon how each case of cylindrical intersection is handled by the application. In doing so it will explain—theoretically and practically—how the GUI interacts with the API, how the API exchanges information with Inventor, as well as how the relevant routines are called within Inventor. Additionally, the purpose, function, and troubleshooting that went into the development of each routine will be analyzed. Finally, the conclusions, the scope for improvement in future work and the general possibilities of Inventor APIs will be mentioned.

Αεί ο θεός γεωμετρεί.

Πλάτων, 427-347 π.Χ., Φιλόσοφος

Μηδείς αγεωμέτρητος εισίτω.

Επιγραφή στην Πλατωνική Ακαδημία

Μηδείς αγεωμέτρητος εισίτω μοι την θύρα»

Η φράση αυτή ανήκει στον Πλάτωνα, ο οποίος προφανώς δεν εννοούσε ότι . . . «απαγορεύει την είσοδο σ΄ όποιον δεν ξέρει γεωμετρία», αλλά ότι απαιτεί απ' τους συνομιλητές του... να έχουν αναλυτικό και επαγωγικό τρόπο σκέψης, αυστηρότητα στην κρίση του και στην διατύπωση συμπερασμάτων, απόλυτη ακρίβεια και σιγουριά στην επιλογή της αλήθειας απ' το ψέμα, του σωστού απ' το λάθος.

Ποιος είναι στ' αλήθεια αγεωμέτρητος; Και πού δεν μπορεί να εισχωρήσει; Αγεωμέτρητος κατά τους αρχαίους Έλληνες φιλοσόφους είναι αυτός που δεν κατέχει επαρκώς τις έννοιες της Δικαιοσύνης, της Ισότητας και της Ακρίβειας. Ο Πλάτωνας πρέσβευε ότι «τα γεωμετρικά σχήματα προϋπάρχουν ως ιδέες και δεν είναι δημιουργήματα του ανθρώπινου πνεύματος». Όσοι λοιπόν είναι αγεωμέτρητοι δεν μπορούν να εισχωρήσουν στις αρχετυπικές τους μνήμες και να ανακαλύψουν τις αρετές που τους διέπουν και τους βοηθούν να εκδηλώσουν τον αληθινό τους εαυτό.

Ευχαριστίες

Το ταξίδι μου στο ΗΜΜΥ σχεδόν τελείωσε και έχω την χαρά αυτή την στιγμή να μπορώ να ευχαριστήσω πρώτα από όλα την οικογένεια μου για όλα τα εφόδια και την στήριξη που 27 χρόνια τώρα μου παρέχουν. Θα ήθελα να αναγνωρίσω το γεγονός ότι από μικρή ηλικία μου έμαθαν να αγαπήσω την γεωμετρία, με την μεταφορική έννοια τουλάχιστον που όπως αναφέρθηκε ο Πλάτωνας της προσδίδει. Μου έμαθαν δηλαδή να έχω αναλυτικό και επαγωγικό τρόπο σκέψης, αυστηρότητα στην κρίση και στην διατύπωση συμπερασμάτων, γεγονός που με έκανε να κατανοήσω και να αγαπήσω τις θετικές επιστήμες, και ως παράγωγο τούτου να βρίσκομαι στην τωρινή θέση να γράφω αυτό το κείμενο. Επιπλέον δεν γίνεται να μην ευχαριστήσω τους φίλους μου και κυρίως αυτούς που μοιραστήκαμε μαζί αυτό το ταξίδι. Όση στήριξη και αν είχαμε από τις οικογένειές μας δεν θα τα καταφέρναμε αν δεν συμπορευόμασταν απέναντι στο απαιτητικό πρόγραμμα σπουδών της σχολής. “Κράτα με να σε κρατώ να ανεβούμε το βουνό”. Έτσι και κάναμε και περάσαμε μαζί αρκετά όμορφα. Θα ήθελα επίσης να ευχαριστήσω τον τελειόφοιτο του ΗΜΜΥ και διδακτορικό στο ΜΠΔ Μαρινάκη Άγγελο για την βοήθειά του όταν αυτή ήταν αναγκαία σε θέματα του Inventor API. Ακόμη οφείλω να ευχαριστήσω θερμά τον κύριο Ζερβάκη Μιχάλη, αντιπρύτανη του πολυτεχνείου, καθηγητή του ΗΜΜΥ και επιβλέπων αυτής της διπλωματικής. Επιπλέον θερμά ευχαριστήρια θα ήθελα να αποδώσω και στον ομότιμο καθηγητή του ΗΜΜΥ κύριο Καλαιτζάκη Κώστα, για την συμμέτοχή του στην εξεταστική επιτροπή. Τέλος θα ήθελα να ευχαριστήσω ιδιαιτέρως τον κύριο Αντωνιάδη Αριστομένη καθηγητή του ΜΠΔ που πέραν της γέννησης της ιδέας αυτής της διπλωματικής και πέραν της συμμετοχής του στην εξεταστική επιτροπή, με βοήθησε αρκετά σε ότι αφορούσε την θεωρία Μηχανολογικού Σχεδίου, μιας και δεν αποτελούσε μέρος των γνωστικών μου πεδίων.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Συντομογραφιών

VBA	Visual Basic for Applications
API	Application Programming Interface
GUI	Graphical User Interface
OOP	Object Oriented Programming
CAD	Computer Aided Design
IDE	Integrated Development Environment
DLL	Dynamic Link Library
COM	Component Object Model

Κεφάλαιο 1

Εισαγωγή

Στην παρούσα διπλωματική υλοποιήθηκε μια διεπαφή προγραμματισμού εφαρμογών, (Application Programming Interface - API). Ένα API επιτρέπει σε μία εφαρμογή να εξάγει πληροφορίες από ένα κομμάτι λογισμικού και να τις χρησιμοποιεί σε μια άλλη εφαρμογή καθώς και επιτρέπει την προγραμματιζόμενη εκτέλεση ενεργειών εντός μίας εφαρμογή, όπως την κλήση ρουτίνων, με σκοπό την αυτοματοποιημένη διευθέτηση εργασιών εντός της.

Το API που υλοποιήθηκε λοιπόν είχε ως σκοπό να καλεί ρουτίνες εντός της εφαρμογής τρισδιάστατης μοντελοποίησης Autodesk Inventor. Υλοποιήθηκε δηλαδή ένα Inventor API. Το εν λόγω API αναπτύχθηκε εντός του Microsoft Visual Studio, το οποίο είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (Integrated Development Environment - IDE). Στο Microsoft Visual Studio δύο ήταν τα βασικά μέρη που αναπτύχθηκαν. Μία διεπαφή χρήστη και οι ρουτίνες όπου παράγουν τα επιθυμητά αντικείμενα εντός του Autodesk Inventor.

Για την ανάπτυξη της διεπαφής χρησιμοποιήθηκε η πλατφόρμα των Windows Forms. Η διεπαφή που δημιουργήθηκε ουσιαστικά είναι ένα GUI (Graphical User Interface) ή αλλιώς ένα Γραφικό Περιβάλλον Χρήστη όπου επιπλέον έχει σαν σκοπό να κρύψει από τον χρήστη (στο background) ότι πραγματικά εκτελείτε εντός του Inventor και μόνο να του παραθέτει τα αποτελέσματα προς αποθήκευση.

Για την υλοποίηση και την εκτέλεση των διαδικασιών εντός του Inventor χρησιμοποιήθηκε η βιβλιοθήκη autodesk.inventor.interop.dll, η οποία είναι μία βιβλιοθήκη δυναμικής σύνδεσης (Dynamic Link Library - DLL). Η γλώσσα που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι η Visual Basic και επομένως η εφαρμογή αυτή μπορεί να θεωρηθεί ένα Inventor VBA (Visual Basic Application).

Να αναφερθεί στο σημείο αυτό ότι στις μέρες μας η δυνατότητα των Inventor APIs είναι ανυπέρβλητη για την αυτοματοποίηση μηχανολογικών εφαρμογών, καθώς ο υπολογισμός και η τρισδιάστατη μοντελοποίηση αναρίθμητων μηχανολογικών εφαρμογών, μπορούν να αυτοματοποιηθούν. Δυστυχώς τα Inventor APIs διέπονται από περίπλοκο και θεωρητικά υπέρογκο οντοκεντρικό προγραμματισμό και επιπλέον απαιτείται η βαθιά κατανόηση του κάθε αντικειμένου του Autodesk Inventor, καθώς και της κάθε μεθόδου που αυτό μπορεί να καλεί.

1.1 Σκοπός της εν λόγω Εργασίας

Σκοπός αυτής της διπλωματικής είναι σε πρώτο βήμα ο υπολογισμός των δισδιάστατων γεωμετρικών αναπτυγμάτων πέντε περιπτώσεων κυλινδρικών τομών. Οι περιπτώσεις αυτές είναι οι εξής:

1. Τομή Κυλίνδρου Με Πλάγιο Επίπεδο
2. Τομή Κυλίνδρου Με Κάθετο Κύλινδρο
3. Τομή Κάθετων Κυλίνδρων Με Απόσταση Ανάμεσα Στους Άξονες
4. Τομή Κυλίνδρων Υπό Κλίση
5. Τομή Κυλίνδρων Υπό Κλίση Και Με Απόσταση Ανάμεσα Στους Άξονες

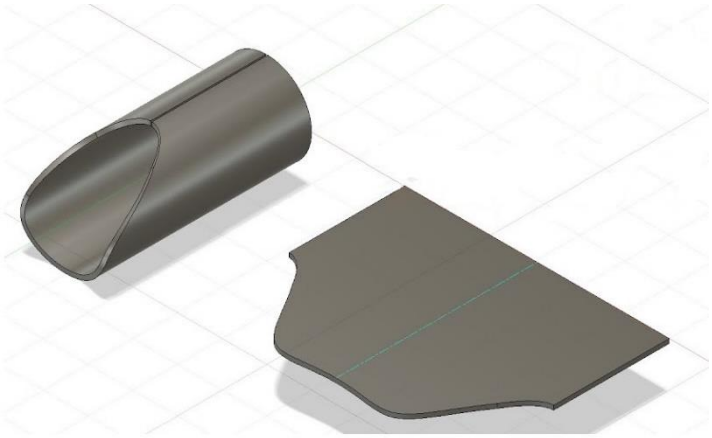
Τα αναπτύγματα των περιπτώσεων αυτών περιέχουν καμπύλες τις οποίες διέπουν πολύπλοκα μαθηματικά μοντέλα και είναι δύσκολο να υπολογισθούν.

Κύριος σκοπός της διπλωματικής εργασίας ήταν λοιπόν η δημιουργία μίας εφαρμογής Inventor VBA η οποία για κάθε μία από τις παραπάνω πέντε περιπτώσεις, και για τις όποιες διαστάσεις δοθούν σαν είσοδο, θα κατασκευάζει αρχικά τα τρισδιάστατα αντικείμενα, έπειτα θα τα αναπτύσσει στον δισδιάστατο χώρο και εν συνέχεια θα υπολογίζει τις καμπύλες που αυτά περιέχουν έπειτα από την ανάπτυξη τους. Αυτό έγινε εφικτό με ρουτίνες που προγραμματίστηκαν να κατασκευάζουν, αναπτύσσουν και υπολογίζουν αντικείμενα εντός του περιβάλλοντος Sheet Metal Part του Autodesk Inventor.

Εν συνέχεια των προαναφερθέντων, αφού το API έχει δημιουργήσει και υπολογίσει τα αναπτύγματα των εν λόγω περιπτώσεων, επιπλέον σκοπός της παρούσας εργασίας είναι και η αυτόματη παραγωγή του μηχανολογικού σχεδίου των κυλινδρικών αναπτυγμάτων για τις όποιες διαστάσεις έχουν δοθεί. Επομένως δημιουργήθηκαν επιπλέον ρουτίνες, που προγραμματίστηκαν να παράγουν το εκάστοτε μηχανολογικό σχέδιο εντός του Drawing περιβάλλοντος του Autodesk Inventor.

1.2 Τα Παραγόμενα και η Χρησιμότητά τους

Η εργασία αυτή λοιπόν υλοποιήθηκε να αυτοματοποιεί τον Υπολογισμό, την Τρισδιάστατη Μοντελοποίηση και την παραγωγή του Μηχανολογικού Σχεδίου των Γεωμετρικών Αναπτυγμάτων των πέντε περιπτώσεων Κυλινδρικών Τομών που ήδη αναφέρθηκαν. Το μηχανολογικό σχέδιο αποφασίστηκε να παράγεται για έως και 50 σημεία ανά καμπύλη, λόγο του περιορισμένου χώρου του φύλλου σχεδιάσεώς εντός του οποίου αυτό περιέχεται. Ωστόσο για περισσότερα από 50 σημεία και έως και για 10000, αποφασίστηκε να υλοποιείται ο Υπολογισμός και η Τρισδιάστατη Μοντελοποίηση, δίχως να υλοποιείται η Παραγωγή του Μηχανολογικού Σχεδίου. Στην εικόνα που ακολουθεί αποφαίνεται η τρισδιάστατη μοντελοποίηση ενός Κυλινδρικού Τμήματος καθώς και του αναπτύγματός του.



ΣΧΗΜΑ 1.1 – ΤΡΙΣΔΙΑΣΤΑΤΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΥΛΙΝΔΡΙΚΟΥ ΤΜΗΜΑΤΟΣ ΚΑΙ ΑΝΑΠΤΥΓΜΑΤΟΣ

Πιθανή χρησιμότητα και εφαρμογές τις παρούσας εργασίας είναι ή μελέτη του μηχανολογικού σχεδίου που παράγεται για μηχανολογικές εφαρμογές. Επιπλέον θα μπορούσε να αποτελέσει εκπαιδευτικό υλικό για την κατανόηση των γεωμετρικών τομών και των αναπτυγμάτων τους, σε εργαστήρια εκμάθησης του Autodesk Inventor. Αυτό αφού κάθε μαθητευόμενος θα μπορεί για τις όποιες διαστάσεις επιθυμεί, μέσα σε ένα λεπτό να παρατηρήσει τα αποτελέσματα και τα αντικείμενα που θα έχουν δημιουργηθεί εντός του Autodesk Inventor. Τρίτον και σημαντικότερο τα αρχεία (.txt) όπως αναφέρθηκαν μπορούν να περιέχουν υπολογισμούς των αναπτυγμάτων για έως και για 10000 σημεία, τα οποία θα μπορούσε να δοθούν ως είσοδο σε μηχανήματα δισδιάστατης κοπής.

Να αναφερθεί ότι τα γεωμετρικά αναπτύγματα των κυλινδρικών τομών συναντώνται συχνά στην κατασκευή σωληνοειδών δικτύων διανομής ρευστών ή αερίων όπως η κατασκευή αεραγωγών εξαερισμού. Συχνό είναι να απαιτούνται πρώτα οι Υπολογισμοί των Αναπτυγμάτων. Αυτό γιατί συχνά σε πρώτο στάδιο είναι βολικό τα μηχανήματα δισδιάστατης κοπής να κόβουν ελάσματα σε μορφή Γεωμετρικών Αναπτυγμάτων και έπειτα τα καταλλήλως κομμένα αυτά ελάσματα να διπλώνονται στον τρισδιάστατο χώρο και να δημιουργούν έτσι τα κατάλληλα τρισδιάστατα κυλινδρικά τμήματα. Ουσιαστικά σε ένα σωληνοειδές δίκτυο το εξάρτημα μίας κυλινδρικής σύνδεσης μπορεί να είναι μία από τις Κυλινδρικές Τομές όπου το API υπολογίζει.

1.3 Περιγραφή των βασικών Κεφαλαίων

Κεφάλαιο 2 – Απαιτούμενο Θεωρητικό Υπόβαθρο: Στο κεφάλαιο αυτό παρουσιάζονται όλες οι απαραίτητες θεωρητικές γνώσεις γύρω από το αντικείμενο της διπλωματικής. Περιγράφεται αναλυτικά ότι ήταν απαραίτητο γύρω από την θεωρία των Γεωμετρικών Αναπτυγμάτων και του μηχανολογικού τους σχεδίου, την τρισδιάστατη μοντελοποίηση, τον προγραμματισμό ενός Inventor API, καθώς και την υλοποίηση ενός GUI μεσώ των Windows Forms.

Κεφάλαιο 3 – Προσέγγιση Και Υλοποίηση: Στο κεφάλαιο αυτό περιγράφονται όλα τα βήματα που ακολουθήθηκαν από το πρώιμο μέχρι και το τελικό στάδιο της εκπόνησης της διπλωματικής εργασίας. Από την κατανόηση δηλαδή του Autodesk Invertor σαν χρήστης του, μέχρι και την αποθήκευση των τελικών παραγομένων από την εφαρμογή που δημιουργήθηκε.

Κεφάλαιο 4 – Αποτελέσματα: Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της διπλωματικής εργασίας. Παρουσιάζονται δηλαδή τα παραγόμενα αρχεία της εφαρμογής Inventor VBA που δημιουργήθηκε, καθώς και τα διαγνωστικά μνήματα και οι πληροφορίες που αυτή παρουσιάζει στον χρήστη μέσω του GUI που υλοποιήθηκε να εμπεριέχει.

Κεφάλαιο 5 – Συμπεράσματα : Στο κεφάλαιο αυτό αναφέρονται τα συμπεράσματα και γενικότερα οι δυνατότητες των Inventor APIs. Επιπλέον περιγράφονται οι περιορισμοί και οι δυσκολίες που προέκυψαν κατά την υλοποίηση της παρούσας εργασίας, καθώς και τα περιθώρια βελτίωσής της σε πιθανή μελλοντική εργασία.

Κεφάλαιο 2

Απαιτούμενο Θεωρητικό Υπόβαθρο

Η συνολική υλοποίηση μπορεί να διαχωριστεί σε τρία μέρη. Τα μέρη αυτά είναι ο προγραμματισμός μιας γραφικής διεπαφής χρήστη, ο προγραμματισμός ενός Inventor API και ο έλεγχος των τιμών εισόδου με τα αντίστοιχα διαγνωστικά μνήματα λάθους. Από άποψη πληροφορικής η γνώση του Αντικειμενοστραφή Προγραμματισμού, ήταν απαραίτητη σε επίπεδο άριστης κατανόησης τόσο για τον προγραμματισμό του Inventor API, όσο και για αυτόν της γραφικής διεπαφής χρήστη. Γενικότερα αρκετά γνωστικά πεδία ήταν απαραίτητα τα οποία και συνολικά αναφέρονται εν συνέχεια.

- Γνώση Αντικειμενοστραφή Προγραμματισμού
- Θεωρία Γεωμετρικών Τομών και Αναπτυγμάτων
- Θεωρία και Κανόνες Μηχανολογικού Σχεδίου
- Γνώσεις Τρισδιάστατης Μοντελοποίησης
- Γνώσεις Γεωμετρίας, Τριγωνομετρίας και Μαθηματικών.

Όλα τα προαναφερθέντα γνωστικά πεδία ήταν απαραίτητα καθ' όλη την διαδικασία εκπόνησης την εν λόγω εργασίας, όχι με ανεξάρτητο, αλλά με συνδυαστικό τρόπο. Στο παρόν κεφάλαιο λοιπόν, θα ακολουθήσει η λεπτομερής περιγραφή όλων αυτών των γνωστικών πεδίων καθώς και η χρησιμότητα αυτών κατά την υλοποίηση της εν λόγω διπλωματικής εργασίας.

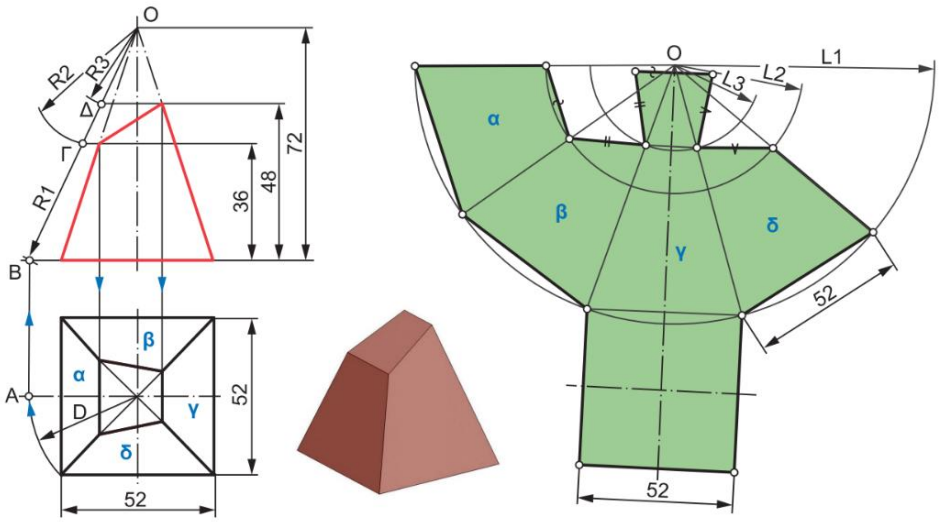
2.1 Θεωρητικό Υπόβαθρο Τρισδιάστατων Αντικειμένων και Γεωμετρικών Αναπτυγμάτων

Τα Γεωμετρικά Αναπτύγματα είναι σχέδια που δείχνουν την επιφάνεια ενός τρισδιάστατου αντικειμένου, ξεδιπλωμένο ή ξετυλιγμένο σε μια δισδιάστατη επίπεδη επιφάνεια (Plane). Να αναφερθεί ότι για κάθε επιφανειακό τρισδιάστατο τμήμα ισοδυναμεί και από ένα ανάπτυγμα. Για παράδειγμα αν μας απασχολεί η γεωμετρική τομή δύο επιφανειακών τρισδιάστατων κυλινδρικών τμημάτων τότε θα πρέπει να υπολογίσουμε δύο γεωμετρικά αναπτύγματα, ένα για κάθε τρισδιάστατο κυλινδρικό τμήμα. Στο σημείο αυτό πρέπει να γίνει σαφές ότι όλα τα τρισδιάστατα αντικείμενα που αναφέρονται παρακάτω θεωρούνται επιφανειακά (Surface Objects).

2.1.1 Τρισδιάστατα Αντικείμενα που Περιέχουν Μόνο Ευθύγραμμα Τμήματα

Τα τρισδιάστατα αντικείμενα που περιέχουν μόνο ευθύγραμμα τμήματα, περιέχουν πολλές γραμμές κάμψης (Bend Lines), γύρω από τις οποίες το τρισδιάστατο αντικείμενο ξεδιπλώνεται προκειμένου να δημιουργηθεί το ανάπτυγμά του. Τέτοια τρισδιάστατα αντικείμενα είναι ο κύβος, οι τριγωνικές πυραμίδες, οι τετραγωνικές πυραμίδες, τα ορθογώνια παραλληλεπίπεδα, τα πρίσματα και όποια άλλα τρισδιάστατα αντικείμενα φέρουν συνδυασμούς αυτών των τρισδιάστατων αντικειμένων.

Όπως απεικονίζεται στο παρακάτω παράδειγμα, τα αναπτύγματα αυτών των τρισδιάστατων αντικειμένων περιέχουν αντίστοιχα μόνο ευθύγραμμα τμήματα και είναι επομένως εύκολο να υπολογιστούν με χρήση απλής τριγωνομετρίας (εφόσον δεν περιέχουν καμπύλες). Στην εικόνα που ακολουθεί λοιπόν απεικονίζεται το μηχανολογικό σχέδιο του αναπτύγματος της τομής μίας Τετραγωνικής Πυραμίδας με ένα Πλάγιο Επίπεδο.



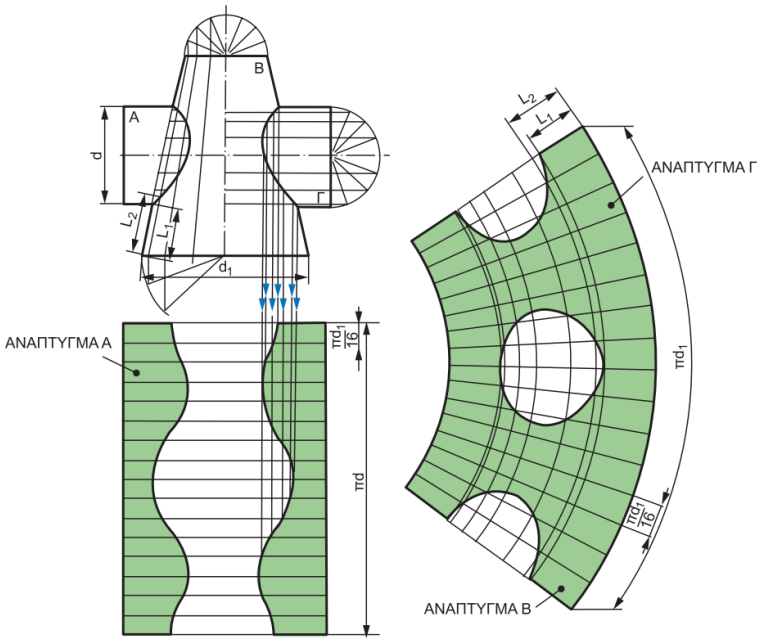
ΣΧΗΜΑ 2.1 – ΑΝΑΠΤΥΓΜΑ ΤΗΣ ΤΟΜΗΣ ΤΕΤΡΑΓΩΝΙΚΗΣ ΠΥΡΑΜΙΔΑΣ ΜΕ ΠΛΑΓΙΟ ΕΠΙΠΕΔΟ

Στην ανωτέρω εικόνα λοιπόν μπορεί να παρατηρηθεί ότι ο υπολογισμός δεκατεσσάρων συγκεκριμένων σημείων είναι πάντα αρκετός για να περιγράψει πλήρως το εν λόγω ανάπτυγμα. Λόγο λοιπόν του εύκολου υπολογισμού τους, τα αναπτύγματα αυτών των τρισδιάστατων αντικειμένων δεν χρίζουν ιδιαίτερης αντιμετώπισης και για τον λόγο αυτό το API που υλοποιήθηκε αποφασίστηκε να μην τα υπολογίζει.

2.1.2 Σφαιρικά, Κυλινδρικά και Κωνικά Τρισδιάστατα Αντικείμενα

Τρισδιάστατα αντικείμενα όπως οι σφαίρες, οι κύλινδροι και οι κώνοι δεν περιέχουν γραμμές κάμψης (Bend Lines) γύρω από τις οποίες αυτά ξεδιπλώνονται, αλλά περιέχουν έναν άξονα συμμετρίας γύρω από τον οποίο αυτά ξετυλίγονται προκειμένου να δημιουργηθούν τα αναπτύγματά τους.

Επιπλέον τέτοια τρισδιάστατα αντικείμενα παρουσιάζουν ιδιαίτερο ενδιαφέρων καθώς δεν περιέχουν μόνο ευθύγραμμα τμήματά αλλά και καμπύλες. Πιο πολύπλοκα τρισδιάστατα συστήματα όπως οι γεωμετρικές τομές δύο οι περισσότερων τέτοιων τρισδιάστατων αντικειμένων περιέχουν καμπύλες που περιγράφονται από πολύπλοκες μαθηματικές τρισδιάστατες συναρτήσεις f(x,y,z). Ακόμα πιο περίπλοκες γίνονται αυτές οι συναρτήσεις όταν αναφερόμαστε σε αναπτύγματα στον δυσδιάστατο χώρο. Αυτό αφού για το ανάπτυγμα του κάθε τέτοιου τρισδιάστατου αντικειμένου, και για κάθε καμπύλή που αυτό περιέχει, θα πρέπει να μετασχηματιστεί η κάθε τρισδιάστατη συνάρτηση καμπύλης f(x,y,z) σε μία νέα δισδιάστατη f(x,y). Η δυσκολία αυτή μπορεί να γίνει και οπτικά αντιληπτή αν αναλογιστούμε δύο τυχαία τέτοια αντικείμενα να τέμνονται. Για παράδειγμα στο παρακάτω σχήμα αποφαίνονται τα αναπτύγματα ενός Κόλουρου Κώνου όπου τέμνετε με έναν Κύλινδρο υπό γωνία 90 μοιρών.



ΣΧΗΜΑ 2.2 – ΑΝΑΠΤΥΓΜΑ ΤΗΣ ΤΟΜΗΣ ΚΟΛΟΥΡΟΥ ΚΩΝΟΥ ΜΕ ΚΑΘΕΤΟ ΚΥΛΙΝΔΡΟ

Όπως μπορεί να γίνει φανερό από την ανωτέρω εικόνα, **καμπύλες περιέχονται, για τις οποίες όσο το δυνατόν περισσότερα σημεία υπολογίζονται, τόσο μεγαλύτερη θα είναι και η ακρίβεια περιγραφής του εκάστοτε μοντέλου**. Πιο συγκεκριμένα οι καμπύλες που περιέχονται τόσο στην τομή των τρισδιάστατων αντικειμένων f(x,y,z), όσο και αυτές που παρουσιάζονται στα αναπτύγματα τους f(x,y), εξαρτώνται από την διάμετρο και το ύψος του Κυλίνδρου καθώς και την διάμετρο, το ύψος και την γωνία κλίσης του Κόλουρου Κώνου. Ακόμα πιο πολύπλοκο θα ήταν αυτό το τρισδιάστατο σύστημα αν η τομή των αντικειμένων δεν γινόταν υπό σταθερή γωνία 90 μοιρών, αλλά αν η γωνία αυτή ήταν μεταβλητή. Τότε και το μέγεθος της γωνίας θα επηρέαζε σε μεγάλο βαθμό τόσο την μορφή των καμπύλων όσο και την μαθηματική τους σχέση. Όχι ολόκληρες οι δισδιάστατες καμπύλες f(x,y), αλλά απλά οι οριακές τιμές των διαστάσεων τους (minX, maxX, minY, maxY), θα ήταν από μόνες τους αρκετά δύσκολο να υπολογιστούν.

Σύμφωνα λοιπών με την ανωτέρω ανάλυση, λόγο της πολυπλεξίας του υπολογισμού των αναπτυγμάτων που φέρουν καμπύλες, αποφασιστικέ η παρούσα διπλωματική εργασία να αποσκοπήσει σε υπολογισμούς τέτοιου τύπου, όπως ο υπολογισμός αναπτυγμάτων για διάφορες περιπτώσεις Κυλινδρικών Γεωμετρικών Τομών.

2.2 Θεωρητικό Υπόβαθρο Μηχανολογικού Σχεδίου

Το Μηχανολογικό Σχέδιο είναι μια τεχνική που χρησιμοποιείται για την αναπαράσταση ενός τρισδιάστατου αντικειμένου σε ένα δισδιάστατο φύλλο σχεδίασης. Τα Μηχανολογικά Σχέδια δίνουν μια ακριβή αναπαράσταση ενός αντικειμένου και αποσκοπώντας σε αυτό περιέχουν μια σειρά από δισδιάστατες όψεις οι οποίες αναπαριστούν όλα τα μέρη με αναλογία μεγέθους (Κλίμακα).

Το Μηχανολογικό Σχέδιο μπορεί να θεωρηθεί ως μία διεθνής τεχνική γλώσσα η οποία επιτρέπει την επικοινωνία μεταξύ των ανθρώπων για την κατασκευή, τον ποιοτικό έλεγχο και τη συναρμολόγηση προϊόντων. Επόμενος διέπεται από ένα σύνολο αυστηρά καθορισμένων κανόνων προκειμένου να μην υπάρχουν περιθώρια αμφισβήτησης των περιεχομένων του.

2.2.1 Μηχανολογικό Σχέδιο Αναπτυγμάτων

Σε ένα μηχανολογικό σχέδιο αναπτυγμάτων, αναπτύγματα παρουσιάζονται εντός ενός φύλου σχεδιάσεως ως δυσδιάστατα σχέδια. Ένα τέτοιο σχέδιο λόγο έλλειψης χώρου σε φύλλο σχεδιάσεως συγκεκριμένων διαστάσεων δεν μπορεί να παρουσιάζεται σε πραγματικές διαστάσεις και επομένως παρουσιάζεται αναλογικά κλιμακωμένο. Γενικότερα σε ένα μηχανολογικό σχέδιο, το περίγραμμα και οι καμπύλες ενός αναπτύγματος πρέπει να ξεχωρίζουν έχοντας πιο παχιές γραμμές με διπλάσιο πάχος από όλες τις υπόλοιπες γραμμές - που για σχεδιαστικούς λόγος τοποθετούνται σε αυτό προκειμένου συνήθως να περιγράφουν διαστάσεις εντός του. Αναλόγως του αν ένα ανάπτυγμα περιέχει μόνο ευθύγραμμα τμήματα ή όχι, διαφέρουν και οι απαιτήσεις του Μηχανολογικού του Σχεδίου.

2.2.1.1 Αναπτύγματα δίχως Καμπύλες

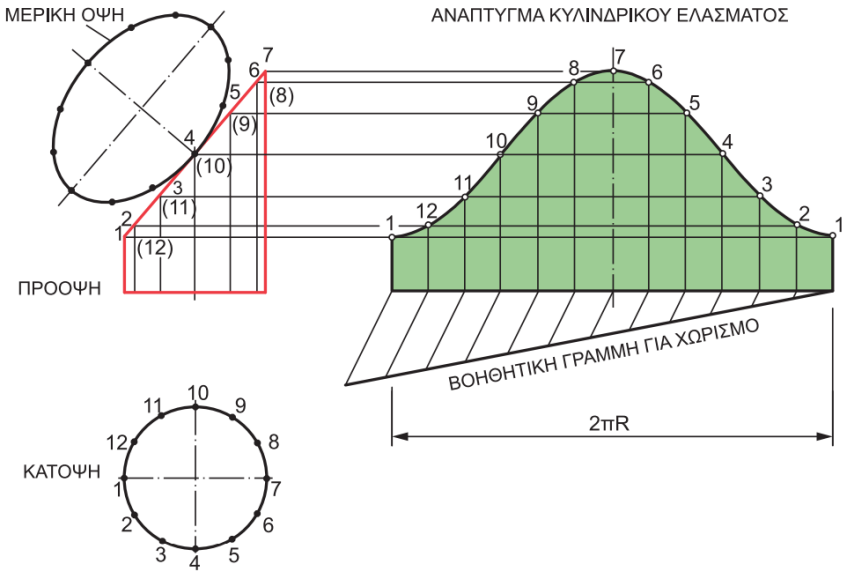
Σε περιπτώσεις όπου ένα ανάπτυγμα περιέχει μόνο ευθύγραμμα τμήματα, δεν απαιτείτε ο υπολογισμός πολλών σημείων για κάθε καμπύλη. Όπως έχει αναφερθεί αρκεί ο υπολογισμός για μικρό πλήθος συγκεκριμένων σημείων, ο οποίος είναι ευκόλως υλοποιήσιμος με χρήση απλής τριγωνομετρίας. Να αναφερθεί επιπλέον ότι σύμφωνα με τους κανόνες του Μηχανολογικού Σχεδίου ένα τέτοιο ανάπτυγμα πέραν από τον κλασικό υπολογισμό των δυσδιάστατων σημείων f(x,y), μπορεί επίσης να ορίζεται πλήρως και με δεύτερο δυνατό τρόπο αν έχουν υπολογιστεί τα μήκη της κάθε ακμής που περιέχει, καθώς και οι γωνίες μεταξύ των ακμών αυτών. Πέραν τούτου δεν απαιτείται περεταίρω περιγραφή για τους κανόνες του Μηχανολογικού Σχεδίου αυτών των αναπτυγμάτων, αφού όπως έχει αναφερθεί το API που δημιουργήθηκε δεν υπολογίζει τέτοια αναπτύγματα δίχως καμπύλες και επομένως μόνο για λόγους σύγκρισης έγινε η παρούσα περιγραφή.

2.2.1.2 Αναπτύγματα που φέρουν Καμπύλες

Σε περιπτώσεις αναπτυγμάτων όπου περιέχουν και καμπύλες είναι απαραίτητος ο υπολογισμών των καμπύλων αυτών. Όπως αναφέρθηκε τα αναπτύγματα αυτά είναι δυσδιάστατα αντικείμενα και επομένως δυσδιάστατες είναι και οι καμπύλες που τα περιγράφουν. Ο υπολογισμός των καμπύλων αυτών δεν μπορεί να γίνει προφανώς για άπειρα δισδιάστατα σημεία (x,y) και έτσι απλά περιορίζεται στον υπολογισμό

ενός μεγάλου πλήθους σημείων για την κάθε καμπύλη. Όσο μεγαλύτερο είναι το πλήθος των σημείων που υπολογίζονται, τόσο καλύτερη είναι και η προσέγγιση (ή περιγραφή) της εκάστοτε καμπύλης. Στο μηχανολογικό σχέδιο λοιπόν τέτοιων αναπτυγμάτων συχνά υπολογίζονται υπεράριθμα σημεία τα οποία και πρέπει να παρουσιάζονται αριθμημένα πάνω στην κάθε καμπύλη.

Στην εικόνα που ακολουθεί παρουσιάζονται σύμφωνα με αυτά που προαναφέρθηκαν, η μορφή και οι πληροφορίες που πρέπει να έχει το μηχανολογικό σχέδιο ενός αναπτύγματος. Το παράδειγμα που ακολουθεί αφορά το μηχανολογικό σχέδιο της Τομής Κυλίνδρου με Πλάγιο Επίπεδο, το οποίο αποτελεί την πρώτη περίπτωση υπολογισμού της εν λόγω διπλωματικής εργασίας.



ΣΧΗΜΑ 2.3 – ΑΝΑΠΤΥΓΜΑ ΤΗΣ ΤΟΜΗΣ ΚΥΛΙΝΔΡΟΥ ΜΕ ΠΛΑΓΙΟ ΕΠΙΠΕΔΟ

Στην ανωτέρω εικόνα δώδεκα σημεία έχουν υπολογιστεί αλλά ωστόσο όπως έχει αναφερθεί όσο περισσότερα σημεία υπολογίζονται, τόσο με μεγαλύτερη ακρίβεια περιγράφεται το εκάστοτε μοντέλο. Από την άλλη, όπως μπορεί να γίνει φανερό από την εν λόγω εικόνα, ένα μηχανολογικό σχέδιο δεν μπορεί να παρουσιάζει πάρα πολλά υπολογισμένα σημεία για λόγους βολικής προβολής και έλλειψης χώρου σε ένα φύλλο σχεδιάσεως περιορισμένων διαστάσεων. Για τον λόγο αυτό όπως και έχει αναφερθεί το API υλοποιήθηκε να παράγει μηχανολογικό σχέδιο που παρουσιάζει υπολογισμούς έως το πολύ 50 σημείων ανά καμπύλη, ενώ για πιο ακριβείς υπολογισμούς με περισσότερα σημεία, παράγεται ένα αρχείο (.txt) εντός του οποίου οι υπολογισμοί έως και 10000 σημείων είναι δυνατόν να περιέχονται.

2.2.2 Γενικοί Κανόνες Μηχανολογικού Σχεδίου

Όπως αναφέρθηκε οι κανόνες που διέπουν ένα μηχανολογικό σχέδιο είναι πολυπληθείς προκειμένου να μην υπάρχουν περιθώρια αμφισβήτησης των περιεχομένων του. Οι γενικοί κανονισμοί μηχανολογικού σχεδίου είναι οι εξής:

- Διαστάσεις φύλλων σχεδιάσεως (DIN 823 Μάιος 1980)
- Δίπλωμα σχεδίων σε μέγεθος A4 (DIN 824, Ιανουάριος 1956)
- Κλίμακες μηχανολογικού σχεδίου (DIN-ISO5455 Δεκέμβριος 1972)
- Καταστάσεις τεμαχίων (DIN 6771 Μέρος 2ο, Σεπτέμβριος 1974)
- Υπόμνημα σχεδίων (DIN 6771 φύλλο 1 Δεκέμβριος 1970)
- Γραμμές σε μηχανολογικά σχέδια: Πάχη, είδη, ομάδες γραμμών και χρησιμοποίησή τους (DIN 15 Φύλλα 1 και 2 Δεκέμβριος 1967)
- Τυποποιημένες μορφές γραμμάτων και αριθμών (DIN 6676 Μέρος 1ο Απρίλιος 1976)

Οι ανωτέρω γενικοί κανονισμοί ήταν όλοι οι απαραίτητοι για την διεκπεραίωση της εν λόγω εργασίας. Ωστόσο απλά να αναφερθεί ότι υπάρχουν πολυάριθμοι επιπλέον κανόνες που μπορεί να διέπουν ένα μηχανολογικό σχέδιο για πολλές διαφορετικές εξειδικευμένες περιπτώσεις. Επομένως επιπλέον κανόνες μπορούν να διέπουν περιπτώσεις όπως:

- Σχεδιαστική παράσταση χαρακτηριστικών στοιχείων μηχανών και χαρακτηριστικών κατασκευαστικών λεπτομερειών
- Σχεδιαστική παράσταση συγκολλήσεων και επικολλήσεων
- Καταχώρηση ανοχών διαστάσεων, γεωμετρικών μορφών και συμβόλων κατεργασιών επιφάνειας σε μηχανολογικά σχέδια

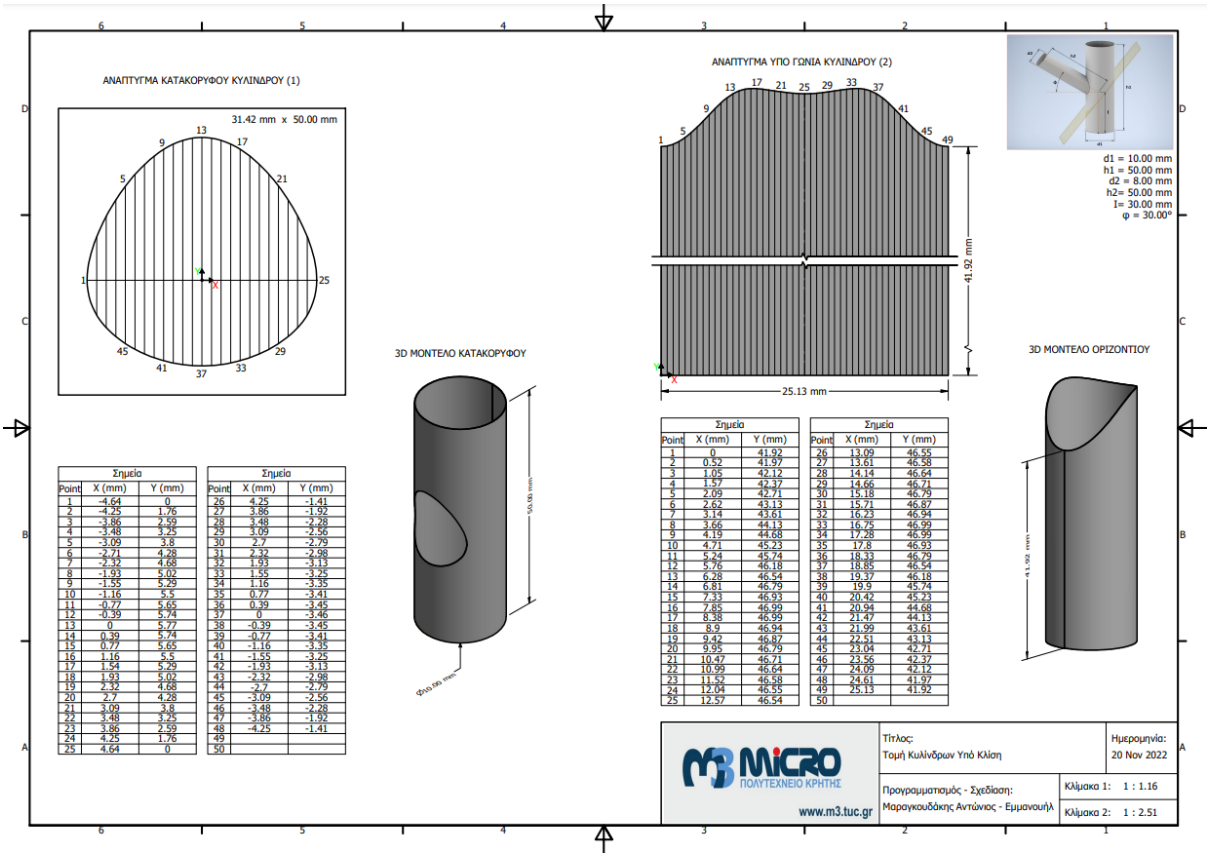
Έπειτα από την γενικότερη αναφορά των κανονισμών του Μηχανολογικού Σχεδίου, στην συνέχεια του παρόντος υποκεφάλαιο θα γίνει η λεπτομερής περιγραφή της εφαρμογής των κανόνων αυτών στα αρχεία Μηχανολογικού Σχέδιου που παράγονται από την εφαρμογή που υλοποιήθηκε. Σύμφωνα με τις παρακάτω περιγραφές εφαρμόστηκαν οι εξής κανόνες:

- Τα αναπτύγματα που παρουσιάζονται αντιπροσωπεύουν μία αναλογία μεγέθους (κλίμακα) των πραγματικών αναπτυγμάτων.
- Οι διάφορες περιγραφές που παρουσιάζονται στο χαρτί του μηχανολογικού σχεδίου μπορούν να περιέχουν κείμενα ή αριθμούς που αυστηρά φέρουν γραμματοσειρά μεγέθους 0.25mm (font size).

- Χρησιμοποιήθηκαν οι διαστάσεις φύλλων σχεδιάσεως Α4 και Α3. Το φύλλο σχεδιάσεως Α4 φέρει διαστάσεις 210mm επί 297mm και χρησιμοποιήθηκε για την πρώτη περίπτωση Τομής Κυλίνδρου με Πλάγιο Επίπεδο. Το φύλλο σχεδιάσεως Α3 φέρει διπλάσια επιφάνια με διαστάσεις 297mm επί 420mm και χρησιμοποιήθηκε για τις υπόλοιπες περιπτώσεις όπου περιέχονται δύο κυλινδρικά τμήματα προς ανάπτυξη και επομένως περισσότερος χώρος απαιτείτε.
- Οι γραμμές των καμπύλων και του περιγράμματος των αναπτυγμάτων είναι παχιές συνεχείς γραμμές μεγέθους 0.7mm. Οι λοιπές γραμμές που σχεδιάζονται εντός του κάθε αναπτύγματος είναι λεπτές συνεχείς γραμμές και ίσες με το μισό των παχών γραμμών, δηλαδή μεγέθους 0.35mm. Οι λεπτές αυτές γραμμές περιέχονται εντός των αναπτυγμάτων προκειμένου να αναδείξουν τις διαστάσεις του κάθε υπολογισμένου σημείου.
- Τα περιγράμματα των αναπτυγμάτων περιέχουν τόσο καμπύλες (όπου υπολογίστηκαν) όσο και ευθύγραμμα τμήματα που αποτελούν τα Κυλινδρικά Πλαίσια. Οι διαστάσεις των Κυλινδρικών πλαισίων προβάλλονται εκτός των αναπτυγμάτων με λεπτές συνεχείς γραμμές.
- Κάποιες φορές οι πραγματικές διαστάσεις του αναπτύγματος που θέλουμε να παρουσιάσουμε στο χαρτί ενός μηχανολογικού σχεδίου είναι αρκετά στενόμακρες, τόσο που δεν επιτρέπουν να είναι ευανάγνωστο ενώ έχει τοποθετηθεί σε περιορισμένο χώρο στο χαρτί. Σε αυτές τις περιπτώσεις μπορεί να κοπεί ένα μέρος του αναπτύγματος αρκεί αυτό να μην περιέχει καμπύλες, αλλά να αποτελεί μέρος μόνο του Κυλινδρικού Πλαισίου. Έτσι κόβοντας περιττά μακρόστενα πλαίσια μπορεί ένα ανάπτυγμα να παρουσιαστεί ευανάγνωστο και με αναλογία κλίμακας στις καμπύλες που περιέχει, με μόνο αντίτιμο την μη αναλογική παρουσίαση των Κυλινδρικών του πλαισίων. Ωστόσο όταν οι καμπύλες που ένα ανάπτυγμα περιέχει είναι από μόνες τους αρκετά στενόμακρες, τότε δεν επιτρέπετε να κοπεί και να μην παρουσιαστεί κομμάτι τους, επομένως αναγκαστικά παρουσιάζεται ένα ποιοτικό ανάπτυγμα με παρόμοια μορφή αλλά δίχως καμία πραγματική αναλογία (κλίμακα).
- Το κάθε ανάπτυγμα αποτελεί ένα δισδιάστατο αντικείμενο και επομένως κάθε σημείο που υπολογίστηκε είναι δισδιάστατο (x,y). Γενικότερα για τον υπολογισμό της κάθε περίπτωσης θεωρήθηκε ένα βολικό σημείο αναφοράς (0,0) κάπου εντός του κάθε αναπτύγματος, το οποίο και θεωρείται κέντρο του συστήματος αξόνων του εκάστοτε αναπτύγματος. Έτσι όλοι οι υπολογισμοί (x,y) έχουν υπολογιστεί σύμφωνα με αντίστοιχο κέντρο αξόνων που έχει θεωρηθεί. Επομένως για κάθε περίπτωση παρουσιάζεται εντός του κάθε αναπτύγματος, το αντίστοιχο σημείο με τη μορφή ενός δυσδιάστατου συστήματος αξόνων.

- Όλα τα υπολογισμένα σημεία πρέπει να φαίνονται στο μηχανολογικό σχέδιο έτσι για κάθε ανάπτυγμα παρουσιάζεται και ένας πίνακας που για κάθε σημείο περιέχει την πληροφορία της απόστασης που φέρει από τον άξονα Χ και αντίστοιχα αυτής που φέρει από τον άξονα Υ. Επιπλέον το κάθε σημείο παρουσιάζεται αριθμημένο στον πίνακα το οποίο και περιγράφεται από τις τιμές Χ και Υ της αντίστοιχης σειράς. Όλες οι πληροφορίες του πίνακα παρουσιάζονται και αυτές με μέγεθος γραμματοσειράς 0.25mm.
- Το κάθε υπολογισμένο σημείο διαφαίνεται επίσης αριθμημένο πάνω στο ανάπτυγμα και εντός της θέσης που αυτό περιγράφει. Η αρίθμηση αυτή ομοίως πραγματοποιείται με μέγεθος γραμματοσειράς 0.25mm. Επιπλέον απαραίτητο είναι η αρίθμηση αυτή των σημείων πάνω στο ανάπτυγμα να αντιστοιχεί με την αρίθμηση που διαφαίνεται στον αντίστοιχο πίνακα που περιέχει τους υπολογισμούς των εν λόγω σημείων.
- Δίπλα στο κάθε ανάπτυγμα παρουσιάζεται και το τρισδιάστατο μοντέλο του αναπτύγματος. Το κάθε τρισδιάστατο μοντέλο παρουσιάζεται προκειμένου να φαίνεται η τρισδιάστατη τομή με πραγματική αναλογία μεγέθους (κλίμακα), χωρίς να παρουσιάζεται το μέγεθος αυτής της κλίμακας εφόσον δεν μας αφορούν οι τρισδιάστατες μετρήσεις ή υπολογισμοί. Επιπλέον το κάθε τρισδιάστατο μοντέλο δεν μας απασχολεί να είναι πλήρως αναλογικό πέραν της τομής που περιέχει, δηλαδή δεν παρουσιάζεται κάποια άλλη πραγματική αναλογία όπως για παράδειγμα το πραγματικά αναλογικό ύψος των κυλίνδρων πάνω ή κάτω από την τομή.
- Κάθε μηχανολογικό σχέδιο στην πάνω δεξιά γωνία του παρουσιάζει επιπλέον μία εικόνα που προσομοιώνει την γενική τρισδιάστατη μορφή της εκάστοτε περίπτωσης Κυλινδρικών Τομών. Στην εν λόγω εικόνα διαφαίνονται ονομασμένες όλες οι απαραίτητες διαστάσεις για την κατασκευή του εκάστοτε τρισδιάστατου μοντέλου. Φυσικά και ο κάθε υπολογισμός πραγματοποιείται για δεδομένες τιμές όπου ένας χρήστης δίνει. Έτσι κάτω από την εικόνα αυτή για την κάθε προβαλλόμενη (απαραίτητη) διάσταση, παρουσιάζονται το σύμβολο της, η δεδομένη τιμή της, καθώς και η αντίστοιχη μονάδα μέτρησής της.
- Τέλος απαραίτητο σε κάθε μηχανολογικό σχέδιο είναι το υπόμνημα. Το υπόμνημα τοποθετείται πάντα στην κάτω δεξιά γωνία του μηχανολογικού σχεδίου και περιέχει απαραίτητες πληροφορίες όπως οι κλίμακες, ο τίτλος του μηχανολογικού σχεδίου, το όνομα του δημιουργού και η ημερομηνία της δημιουργίας. Ένα υπόμνημα υλοποιήθηκε να φέρει πάντα διαστάσεις 180mm επί 36mm.

Όλοι οι κανόνες που αναφέρθηκαν αποφαίνονται και μπορούν να γίνουν κατανοητοί στην εικόνα Μηχανολογικού Σχεδίου που ακολουθεί. Το εν λόγω μηχανολογικό σχέδιο είναι ένα παραγόμενο PDF της εφαρμογής που δημιουργήθηκε και αφορά την 4η περίπτωση υπολογισμού (Τομή Κυλίνδρων Υπό Κλίση).



ΣΧΗΜΑ 2.4 – ΠΑΡΑΓΟΜΕΝΟ ΜΗΧΑΝΟΛΟΓΙΚΟ ΣΧΕΔΙΟ ΤΗΣ 4ΗΣ ΠΕΡΙΠΤΩΣΗΣ ΚΥΛΙΝΔΡΙΚΩΝ ΤΟΜΩΝ

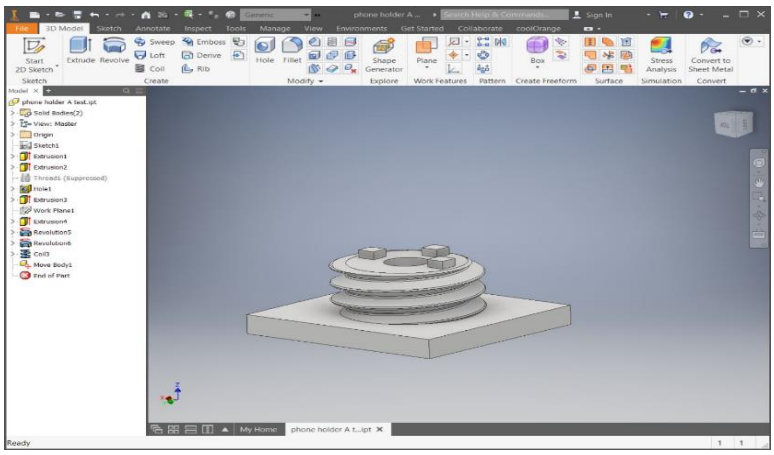
2.3 Πλατφόρμα Autodesk Inventor

Το Autodesk Inventor είναι μία εφαρμογή που απευθύνεται κυρίως σε μηχανολόγους και επιτρέπει στους χρήστες να κατασκευάζουν ακριβή τρισδιάστατα μοντέλα που αποσκοπούν στην προσομοίωση και την οπτικοποίηση πριν από την κατασκευή προϊόντων. Το Autodesk Inventor είναι μια εφαρμογή τύπου CAD (Computer Aided Design - Σχεδίαση με τη βοήθεια υπολογιστή). Οι εφαρμογές CAD παράγουν και αποθηκεύουν αρχεία όπου περιέχουν δισδιάστατες ή τρισδιάστατες σχεδιάσεις. Το Autodesk Inventor δημιουργεί και αποθηκεύει τέσσερίς διαφορετικούς τύπους αρχείων (όπως .ipt, .iam, .dwg και .ipn) και εκτελείται σε λειτουργικά συστήματα Windows και Mac.

Το Inventor λοιπόν υποστηρίζει τέσσερα διαφορετικά περιβάλλοντα εντός τον οποίων μπορούν να δημιουργηθούν διαφορετικά αρχεία των τύπων που αναφέρθηκαν. Κάθε τέτοιο περιβάλλον υποστηρίζει διαφορετικές ενέργειες και χρησιμοποιείται για διαφορετικούς λόγους όπως περιγράφεται παρακάτω.

Part Files (.ipt)

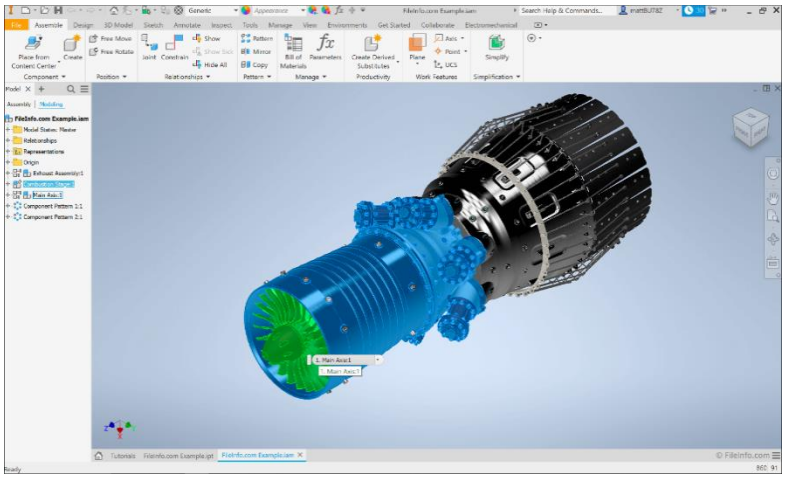
Το περιβάλλον όπου δημιουργούνται αρχεία τύπου Part μπορεί να θεωρηθεί ένα περιβάλλον σχεδιασμού ενός εξαρτήματος. Οι εντολές – ενέργειες που υποστηρίζονται σε αυτό το περιβάλλον δημιουργούν αρχικά δισδιάστατα σκίτσα (2D Sketch). Εφόσον τα σκίτσα περιέχουν κλειστές επιφάνειες (Planes), μπορούν με κατάλληλες ενέργειες να αποκτήσουν διαστάσεις στον χώρο προκειμένου να δημιουργηθούν τρισδιάστατα αντικείμενα. Ένα σκίτσο μπορεί να διέπεται από οποιαδήποτε γεωμετρία. Γενικότερα, πολυάριθμες ενέργειες μπορούν να εφαρμοστούν πάνω σε ένα σκίτσο προκειμένου να δημιουργηθεί ένα οποιοδήποτε τρισδιάστατο αντικείμενο το οποίο πολλές ακόμα ενέργειες μπορούν να παραλλάξουν. Επιπλέον κάθε τρισδιάστατο αντικείμενο εντός του περιβάλλοντος Part μπορεί να ενωθεί με άλλα ή να δημιουργήσει την τομή αυτού με κάποιο άλλο τρισδιάστατο αντικείμενο ή επίπεδο (Plane). Έτσι με συνδυασμούς των παραπάνω ενεργειών μπορούν να δημιουργηθούν τρισδιάστατα αντικείμενα (μοντέλα) αυξημένης πολυπλοκότητας. Στην εικόνα που ακολουθεί αποφαίνεται το περιβάλλον Part της εφαρμογής Autodesk Inventor.



ΣΧΗΜΑ 2.5 – PART ΠΕΡΙΒΑΛΛΟΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ AUTODESK INVENTOR

Assembly Files (.iam)

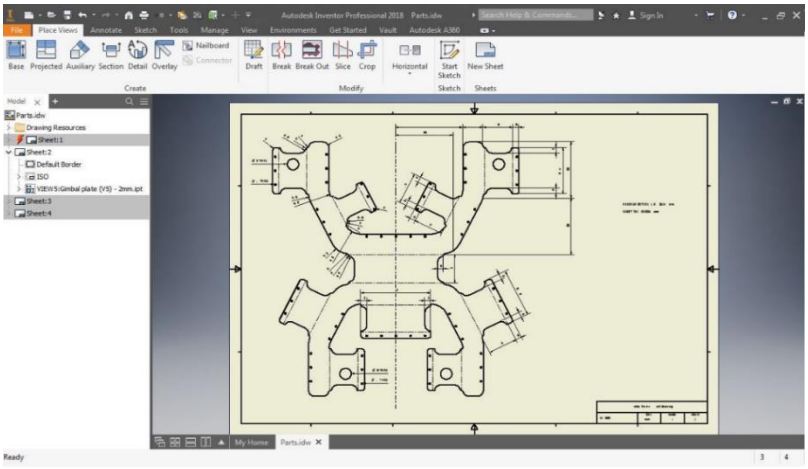
Το περιβάλλον όπου δημιουργούνται αρχεία τύπου Assembly είναι ένα περιβάλλον συναρμολόγησης πολλών εξαρτημάτων (Part), με σκοπό την μοντελοποίηση ενός ολοκληρωμένου προϊόντος. Οι ενέργειες αυτού του περιβάλλοντος επιτρέπουν την ομαδοποίηση εξαρτημάτων που λειτουργούν μαζί ως ενιαία μονάδα. Μία τέτοια ομάδα σε επόμενο βήμα μπορεί να θεωρηθεί υποσυγκρότημα σε άλλο συγκρότημα (νέα μεγαλύτερη ενιαία μονάδα). Γενικότερα, υπάρχουν περιορισμοί συναρμολόγησης μεταξύ των εξαρτημάτων οι οποίοι καθορίζουν τη θέση και τη συμπεριφορά τους. Με κατάλληλες ενέργειες μπορούν να δοθούν αυστηροί βαθμοί ελευθερίας κινήσεων στο χώρο για το κάθε εξάρτημα. Συνεπώς τα εξαρτήματα που ανήκουν σε ένα αρχείο Assembly μπορούν να υλοποιηθούν ώστε να συμπεριφέρονται όπως θα έκαναν στον φυσικό κόσμο. Στην εικόνα που ακολουθεί αποφαίνεται το περιβάλλον Assembly της εφαρμογής Autodesk Inventor.



ΣΧΗΜΑ 2.6 – ASSEMBLY ΠΕΡΙΒΑΛΛΟΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ AUTODESK INVENTOR

Drawing Files (.idw, .dwg)

Το περιβάλλον όπου δημιουργούνται αρχεία τύπου Drawing υποστηρίζει ενεργείς οι οποίες επιτρέπουν την δημιουργία ενός Μηχανολογικού Σχεδίου για τα οποιαδήποτε εξαρτήματα (.ipt Files) ή συγκροτήματα εξαρτημάτων (.iam Files), έχουν ήδη δημιουργηθεί. Σε ένα Drawing File είναι εφικτή η τοποθέτηση όψεων ενός μοντέλου σε ένα ή περισσότερα φύλλα σχεδίασης. Στη συνέχεια είναι δυνατόν να γίνει προσθήκη διαστάσεων και άλλων σχολιασμών σχεδίασης με σκοπό την τεκμηρίωση του μοντέλου. Ένα σχέδιο που τεκμηριώνει ένα συγκρότημα εξαρτημάτων (.iam File), μπορεί να περιέχει μια αυτοματοποιημένη λίστα εξαρτημάτων εκτός από τις απαιτούμενες προβολές όψεων. Ένα αρχείο Drawing διατηρεί συνδέσμους μεταξύ των αρχείων (.ipt και .iam) που προβάλει. Έτσι είναι δυνατή ή τροποποίηση ενός αρχείου Drawing οποιαδήποτε στιγμή έπειτα από την τροποποίηση ενός αντίστοιχου συνδεδεμένου αρχείου. Από προεπιλογή δηλαδή, ένα σχέδιο ενημερώνεται αυτόματα μετά από την επεξεργασία ενός στοιχείου του (.ipt, .iam). Στην εικόνα που ακολουθεί αποφαίνεται το περιβάλλον Drawing της εφαρμογής Inventor.

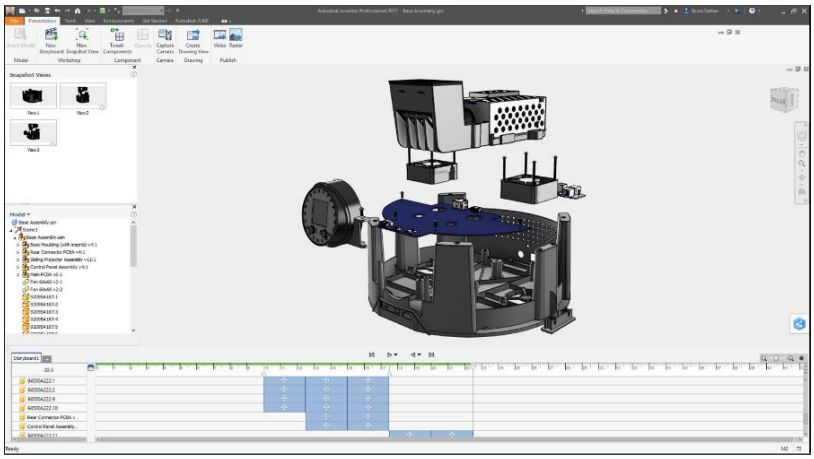


ΣΧΗΜΑ 2.7 – DRAWING ΠΕΡΙΒΑΛΛΟΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ AUTODESK INVENTOR

Presentation Files (.ipn)

Το περιβάλλον αυτό υποστηρίζει ενέργειες που βοηθούν στην δημιουργία ενός αρχείου χρήσιμου για την παρουσίαση ενός προϊόντος. Τα αρχεία παρουσίασης (Presentation Files) είναι ένας τύπος αρχείου που επιτρέπει την παρουσίαση οποιασδήποτε σχεδίασης έχει ήδη πραγματοποιηθεί σε οποιοδήποτε από τρία προαναφερθέντα περιβάλλοντα (Part, Assembly, Drawing). Για παράδειγμα στο περιβάλλον αυτό μπορεί να δημιουργηθεί μια διευρυμένη προβολή μιας διάταξης εξαρτημάτων (.ipt αρχείων), η οποία με χρήση κινούμενων σχεδίων είναι δυνατόν να δείχνει τη σειρά συναρμολόγησης βήμα προς βήμα

ενός αρχείου συναρμολόγησης (.iam). Ένα κινούμενο σχέδιο μπορεί να περιέχει διάφορες γωνίες προβολής και την κατάσταση ορατότητας των στοιχείων του κάθε βήματος της διαδικασίας συναρμολόγησης. Τα εν λόγω κινούμενα σχέδια μπορούν να αποθηκευτούν με μορφή αρχείου .wmv ή .avi. Στην εικόνα που ακολουθεί αποφαίνεται το περιβάλλον Presentation της εφαρμογής Autodesk Inventor.



ΣΧΗΜΑ 2.8 – PRESENTATION ΠΕΡΙΒΑΛΛΟΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ AUTODESK INVENTOR

Απαραίτητη ήταν λοιπόν η άριστη γνώση της εφαρμογής Autodesk Inventor εφόσον το API έπρεπε να εκτελεί ενέργειες και να δρομολογεί διαδικασίες εντός των περιβάλλοντων Part και Drawing. Στο περιβάλλον τύπου Part το API ενεργεί με σκοπό την αυτοματοποιημένη παραγωγή των τρισδιάστατων μοντέλων των Κυλινδρικών Τομών και έπειτα χρησιμοποιώντας αυτά τα μοντέλα, ενεργεί με σκοπό την παραγωγή των αναπτυγμάτων τους. Στο περιβάλλον τύπου Drawing το API ενεργεί με σκοπό την παραγωγή του Μηχανολογικού σχεδίου των αναπτυγμάτων που ήδη έχει δημιουργήσει και αποθηκεύσει στο περιβάλλον τύπου Part (.ipt File). Η κατανόηση των υπόλοιπων δύο περιβάλλοντων δεν ήταν απαραίτητη εφόσον το API δεν χρειάστηκε να επενεργεί εντός τους και συνεπώς αναφέρθηκαν μόνο για λόγους πληρότητας της περιγραφής του Inventor.

2.3.1 Τρισδιάστατη Μοντελοποίηση στο Inventor Part

Όπως αναφέρθηκε απαραίτητη ήταν η τρισδιάστατη μοντελοποίηση των Κυλινδρικών Τομών. Για να γίνει αυτό εφικτό αναγκαία ήταν η άπταιστη γνώση τις ιεραρχίας των αντικειμένων που το Inventor Part διαχειρίζεται καθώς και των διαθέσιμων ενεργειών που το κάθε αντικείμενο επιτρέπει (μεθόδων).

Σύμφωνα με τα προαναφερθέντα, για την δημιουργία ενός οποιοδήποτε τρισδιάστατου αντικειμένου αρχικά δημιουργούνται κατάλληλα δισδιάστατα σκίτσα (2D Sketch), τα οποία πρέπει να περιέχουν κλειστές επιφάνειες (Planes) όπου με κατάλληλες ενέργειες μπορούν να αποκτήσουν διαστάσεις στον χώρο και να δημιουργήσουν τρισδιάστατα αντικείμενα. Αφού έχει παραχθεί ένα όποιο τρισδιάστατο αντικείμενο επιπλέον ενέργειες μπορούν να εφαρμοστούν σε αυτό, οι οποίες έχουν την δυνατότητα να το παραλλάξουν, να το ενώσουν με άλλα ή να δημιουργήσουν την τομή αυτού με κάποιο άλλο τρισδιάστατο αντικείμενο ή επίπεδο (Plane). Κατά αυτόν τον τρόπο λοιπόν κατάλληλες ενέργειες μπορούν να δημιουργήσουν τα τρισδιάστατα μοντέλα των Κυλινδρικών Τομών τα οποία αποτελούν απαραίτητη προεργασία ώστε σε επόμενο βήμα να επιτευχθεί η δημιουργία των δισδιάστατων αναπτυγμάτων τους.

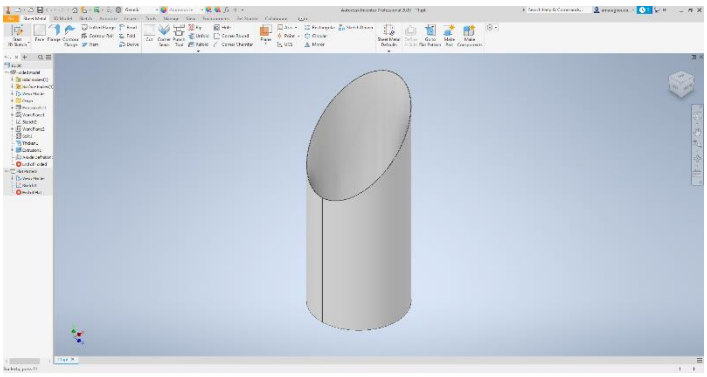
Προκειμένου να προχωρήσουμε στην δισδιάστατη ανάπτυξη των τρισδιάστατων μοντέλων, απαραίτητο είναι το αρχείο τύπου Part να φέρει ως δευτερεύοντα τύπο (Subtype) το Sheet Metal. Ο εν λόγω δευτερεύοντας τύπος προσδίδει μερικές ακόμα δυνατότητες στο περιβάλλον τύπου Part. Οι δυνατότητες αυτές δεν είναι άλλες από μερικές ακόμα ενέργειες (μέθοδοι) που επιτρέπουν την ανάπτυξη ενός τρισδιάστατου μοντέλου στον δισδιάστατο χώρο. Επομένως πλέον μπορεί να γίνεται λόγος για Sheet Metal Part.

Αξίζει να αναφερθεί ότι ένα Sheet Metal Part επιτρέπει την ανάπτυξη τρισδιάστατων αντικειμένων τα οποία θεωρούνται ότι έχουν δημιουργηθεί από ένα επίπεδο κομμάτι μετάλλου με σταθερό πάχος (έλασμα). Δηλαδή τόσο τα τρισδιάστατα μοντέλα όσο και τα αναπτύγματα τους περιέχουν την έννοια του σταθερού πάχους (Thickness). Πιο συγκεκριμένα το περιβάλλον Sheet Metal Part ενσωματώνει ένα σύνολο κανόνων όπου καθορίζουν ορισμένα κοινά χαρακτηριστικά όπως το πάχος του υλικού, οι κανόνες ξεδίπλωσης, η σταθερά ευλυγισίας του υλικού και λοιπά.

Στο περιβάλλον Sheet Metal Part, εφόσον ένα τρισδιάστατο μοντέλο πληροί όλες τις προϋποθέσεις για να αναπτυχθεί στον δισδιάστατο χώρο, αυτό μπορεί εύκολα να επιτευχθεί με την χρήση της μεθόδου Unfold. Αφού λοιπόν αυτό το γεγονός έχει συμβεί, ένα επιπλέον αντικείμενο τύπου Flatt Pattern θα έχει δημιουργηθεί, το οποίο θα περιέχει το ανάπτυγμα ξαπλωμένο στον τρισδιάστατο χώρο.

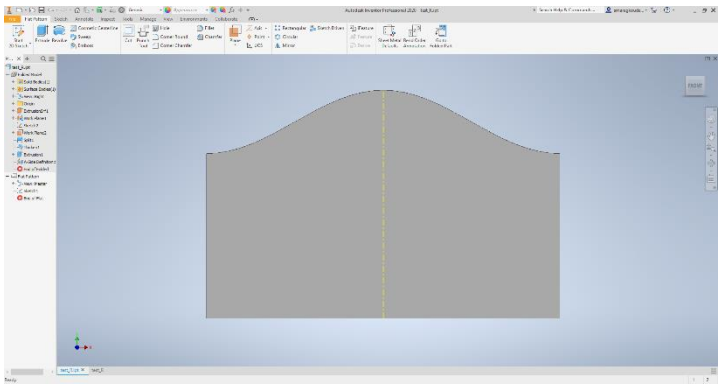
Να αναφερθεί ότι οι προϋποθέσεις που πρέπει να πληροί ένα τρισδιάστατο μοντέλο για να αναπτυχθεί στον δισδιάστατο χώρο διαφέρουν ανάλογα με τον τύπο του τρισδιάστατου μοντέλου όπως αυτοί περιγράφηκαν. Σε περίπτωση που το τρισδιάστατο μοντέλο φέρει μόνο ευθύγραμμές ακμές, απαραίτητο είναι να έχουν δημιουργηθεί σωστά όλες οι ακμές του ως γραμμές κάμψης (Bend Lines) και να έχουν προσδιοριστεί οι κανόνες ξεδίπλωσης. Διαφορετικά σε περίπτωση όπου το τρισδιάστατο αντικείμενο φέρει καμπύλες κλειστού

τύπου (Κύλινδρος, Κώνος), απαραίτητο είναι στο τρισδιάστατο μοντέλο αυτό να έχει εφαρμοστεί μία σχισμή (Rip) παράλληλα στον άξονα συμμετρίας του, γύρω από τον οποίο και πρόκειται να αναπτυχθεί – ξετυλιχθεί. Η εφαρμογή μίας τέτοιας σχισμής επιτρέπει την ανάπτυξη ενός τέτοιου τρισδιάστατου μοντέλου αφού οι καμπύλες κλειστού τύπου που πριν περιείχε παύουν να είναι κλειστού τύπου, αλλά πλέον παρουσιάζουν αρχή και τέλος. Στην εικόνα που ακολουθεί αποφαίνεται εντός του περιβάλλοντος Part ένα Κυλινδρικό Τμήμα που περιέχει μια κατάλληλη σχισμή ως απαραίτητο κανόνα ξεδίπλωσης.



ΣΧΗΜΑ 2.9 – ΚΥΛΙΝΔΡΙΚΟ ΤΜΗΜΑ ΜΕ ΚΑΤΑΛΛΗΛΗ ΣΧΙΣΜΗ, ΕΝΤΟΣ ΤΟΥ PART ΠΕΡΙΒΑΛΛΟΝΤΟΣ

Έπειτα από το πέρας της διαδικασίας που περιγράφηκε ένα αντικείμενο τύπου Flatt Pattern θα έχει πλέον δημιουργηθεί, το οποίο θα περιέχει όπως αναφέρθηκε το ανάπτυγμα ξαπλωμένο στον τρισδιάστατο χώρο. Ακόμα λοιπόν το μοντέλο του αναπτύγματος θα παρευρίσκετε στον τρισδιάστατο εικονικό χώρο του Inventor Part, με την διάσταση του πάχους του να έχει τον ρόλο της τρίτης διάστασης. Ωστόσο στον δισδιάστατο υπολογισμό των καμπύλων που ακολουθεί, η τρίτη διάσταση (του πάχους) μπορεί να αγνοηθεί αφού δεν επηρεάζει τις καμπύλες των αναπτυγμάτων οι οποίες εξελίσσονται μόνο στις διαστάσεις μήκους και πλάτους. Στην εικόνα που ακολουθεί αποφαίνεται το αντικείμενο Flatt Pattern (ανάπτυγμα) του Κυλινδρικού Τμήματος της προηγούμενης φωτογραφίας.



ΣΧΗΜΑ 2.10 – ΑΝΑΠΤΥΓΜΑ ΚΥΛΙΝΔΡΙΚΟΥ ΤΜΗΜΑΤΟΣ ΕΝΤΟΣ ΤΟΥ PART ΠΕΡΙΒΑΛΛΟΝΤΟΣ

Για τον υπολογισμό της κάθε καμπύλης δεν ήταν απαραίτητη η γνώση μαθηματικών. Αυτό γιατί το Inventor περιέχει όλη την απαραίτητη τρισδιάστατη μαθηματική λογική προκειμένου να μπορεί να μοντελοποιεί τρισδιάστατα αντικείμενα.

Τέλος, να αναφερθεί ακόμα ότι οποιοδήποτε αντικείμενο παράγετε στον εικονικό τρισδιάστατο χώρο του Inventor περιέχει ένα σετ υπό-αντικειμένων και πληροφοριών που το χαρακτηρίζουν. Για παράδειγμα ένα τρισδιάστατο επιφανειακό αντικείμενο περιέχει πολλά επιφανειακά σώματα (SurfaceBodys) τα οποία με την σειρά τους περιέχουν πολλά πρόσωπα (Faces) τα οποία με την σειρά τους περιέχουν πολλές γραμμές (Edges). Κάθε τέτοιο υπό-αντικείμενο περιέχει επίσης και ένα σετ πληροφοριών που το χαρακτηρίζει. Να αναφερθεί ότι κάθε ενέργεια που τροποποιεί ένα τρισδιάστατο αντικείμενο αυτόματος επανα-υπολογίζει όλες τις πληροφορίες που χαρακτηρίζουν αυτό και όλα τα υπό-αντικείμενά του. Για το Inventor κάθε καμπύλη είναι ένα αντικείμενο τύπου (Edge) το οποίο περιέχει ένα σετ πληροφοριών όπου και την χαρακτηρίζουν. Έτσι με κατάλληλο κώδικα είναι εφικτή η άντληση της πληροφορίας (x,y) οποιουδήποτε σημείου μία καμπύλη περιέχει, αφού η πληροφορία αυτή περιέχεται εντός ενός αντικειμένου τύπου Edge.

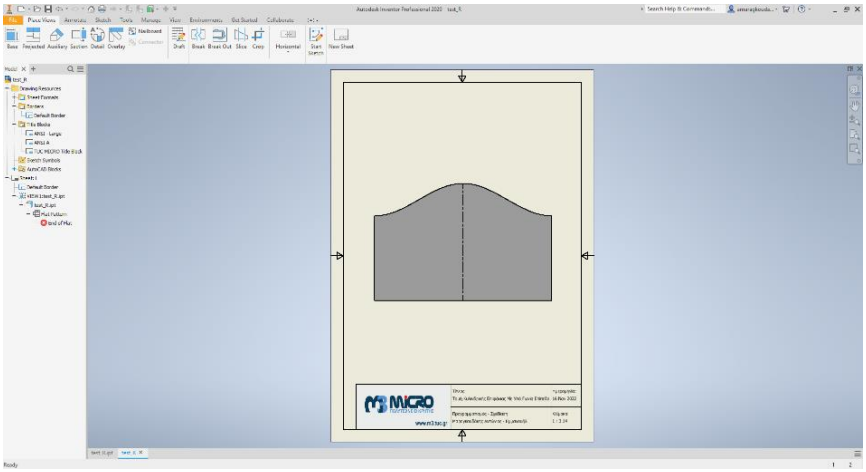
Κλείνοντας λοιπόν, θα μπορούσαμε να πούμε ότι δεν υπολογίσαμε τα εν λόγω σημεία αλλά χορέψαμε πλάι στην μαθηματική ισχύ του Inventor το οποίο τα υπολόγισε για εμάς μέσω των τρισδιάστατων αντικειμένων, τα οποία και εν συνέχεια υλοποιήθηκε να προβάλλονται εντός του Μηχανολογικού Σχεδίου που τελικά παράγεται όπως θα περιγραφεί παρακάτω.

2.3.2 Παραγωγή Μηχανολογικού Σχεδίου στο Inventor Drawing

Όπως έχει αναφερθεί το περιβάλλον τύπου Drawing υποστηρίζει ενέργειες οι οποίες επιτρέπουν την δημιουργία ενός Μηχανολογικού Σχεδίου για οποιαδήποτε εξαρτήματα (.ipt Files) ή συγκροτήματα εξαρτημάτων (.iam Files) έχουν ήδη μοντελοποιηθεί. Επομένως έχοντας ολοκληρώσει την τρισδιάστατη μοντελοποίηση και ανάπτυξη που περιγραφικέ εντός αποθηκευμένου αρχείου Part (.ipt), είναι πλέων δυνατόν στο περιβάλλον του Drawing να αντληθεί η οποιαδήποτε όψη. Συνεπώς το δισδιάστατο περίγραμμα ενός τρισδιάστατου αναπτύγματος μπορεί να προβληθεί εντός ενός οποιουδήποτε φύλου σχεδιάσεως επιτρεπτών διαστάσεων (π.χ. A3, A4, κλπ.).

Προφανώς το περίγραμμα του αναπτύγματος στο περιβάλλον του Drawing είναι και αυτό ένα νέο αντικείμενο που με την σειρά του περιέχει υπό-αντικείμενα και σετ πληροφοριών. Να αναφερθεί ότι το αντικείμενο αυτό αντλεί τις πληροφορίες του από το τρισδιάστατο

αντικείμενο αναπτύγματος που δημιουργήθηκε στο Part περιβάλλον το οποίο περιείχε και διάσταση πάχους ως τρίτη διάσταση. Ωστόσο το αντικείμενο που παρουσιάζεται στο Drawing είναι μία δισδιάστατη όψη αναπτύγματος. Είναι δηλαδή ένα νέο αντικείμενο που είναι αυστηρά δισδιάστατο και μπορούμε να το φανταστούμε σαν μία κάτοψη του προκάτοχου τρισδιάστατου αντικειμένου του. Στην εικόνα που ακολουθεί αποφαίνεται η όψη ενός αναπτύγματος ενός κυλινδρικού τμήματος εντός του περιβάλλοντος Drawing.



ΣΧΗΜΑ 2.11 – ΠΕΡΙΓΡΑΜΜΑ ΑΝΑΠΤΥΓΜΑΤΟΣ ΕΝΤΟΣ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ DRAWING

Το δισδιάστατο ανάπτυγμα που παρουσιάζεται στο Drawing θα πρέπει να καταλαμβάνει συγκεκριμένο χώρο στο φύλο σχεδιάσεως και αυτό επιτυγχάνεται με τον υπολογισμό της κατάλληλης κλίμακας που θα το διαστασιοποιεί. Όταν το ανάπτυγμα είναι αρκετά στενόμακρο, παρουσιάζονται διάφορα προβλήματα που αφορούν την τοποθέτησή του στο χώρο του φύλλου σχεδιάσεως. Τα προβλήματα αυτά όπως και η αντιμετώπιση τους θα αναφερθούν στο επόμενο κεφάλαιο.

Πέραν της τοποθέτησης του δυσδιάστατου αναπτύγματος στο φύλο σχεδιάσεως, υπεράριθμες ακόμα ενέργειες μπορούν να πραγματοποιηθούν στο περιβάλλον του Drawing προκειμένου να δημιουργηθούν όλα τα απαραίτητα στοιχεία που ένα Μηχανολογικό Σχέδιο πρέπει να περιέχει σύμφωνα με τους κανόνες που αναφέρθηκαν να το διέπουν. Δηλαδή διαστάσεις ακμών, πίνακες με σημεία, κείμενα, υπόμνημα και άλλα μπορούν να παρουσιαστούν στο περιβάλλον του Drawing ώσπου τελικά να δημιουργηθεί ένα πλήρες και αυστηρά ορισμένο Μηχανολογικό Σχέδιο. Αφού όλα τα απαραίτητα υλοποιηθούν, είναι δυνατή η αποθήκευση του Μηχανολογικού Σχεδίου και σε μορφή αρχείου PDF, πέραν της αποθήκευσης με μορφή αρχείου Drawing (.idw ή .dwg). Ένα τέτοιο παραγόμενο μηχανολογικό σχέδιο έχει ήδη παρατεθεί στο ΣΧΗΜΑ 2.4.

2.4 Προγραμματισμός Inventor API

Πολλά λογισμικά CAD εκθέτουν το API τους για να επεκτείνουν τη βασική τους λειτουργικότητα, έτσι ώστε να μπορούν να αναπτυχθούν πρόσθετα λογισμικά με σκοπό να βοηθούν κάποιους χρήστες να αυξήσουν την παραγωγικότητά τους. Το Autodesk Inventor εκθέτει το API του με τη μορφή ενός μοντέλου ιεραρχίας αντικειμένων (Component Object Model – COM). Δεδομένου ότι το API είναι βασισμένο σε αυτό το αντικειμενοστραφές μοντέλο, ομοίως απαιτείται και ο προγραμματισμός του σε γλώσσα που να υποστηρίζει αντικειμενοστραφή προγραμματισμό. Σχεδόν οποιαδήποτε δημοφιλή γλώσσα προγραμματισμού μπορεί να προγραμματίσει ένα Inventor API, όπως Visual Basic, C++, C#, Delphi, Python και Java. Η γλώσσα προγραμματισμού που επιλέχτηκε για το API που δημιουργήθηκε είναι η Visual Basic και επομένως η εν λόγω εφαρμογή μπορεί να χαρακτηριστεί και ως ένα Inventor VBA (Visual Basic Application). Επιλέχτηκε λοιπόν η Visual Basic αφενός γιατί υπάρχει περισσότερο υποστηρικτικό υλικό και αφετέρου γιατί ήταν βολικό ταυτόχρονα στο Visual Studio να υλοποιηθεί και ο προγραμματισμός του API αλλά και των γραφικών (GUI) όπου θα περιγραφεί στο επόμενο υπο-κεφάλαιο 2.5.

2.4.1 Αντικειμενοστραφής Προγραμματισμός

Ο αντικειμενοστραφής προγραμματισμός (Object Oriented Programming - OOP) είναι μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο χειρισμός των σχετιζόμενων δεδομένων και των διαδικασιών γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Μία τέτοια δομή δεδομένων καλείται αντικείμενο και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων που ονομάζεται κλάση (Class). Η κλάση προδιαγράφει τόσο τα δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους.

Επιπλέον ο αντικειμενοστραφής προγραμματισμός εμπεριέχει την έννοια της κληρονομικότητας μεταξύ αντικειμένων. Ένα αντικείμενο μπορεί έχει υπο-αντικείμενα ενώ ταυτόχρονα μπορεί να είναι υπο-αντικείμενο ενός άλλου αντικειμένου και κατά αυτόν τον τρόπο αναπτύσσονται σχέσεις κληρονομικότητας μεταξύ των αντικειμένων και υπο-αντικειμένων. Οτιδήποτε δημιουργείτε εντός του Autodesk Inventor σε επίπεδο προγραμματισμού είναι ένα αντικείμενο το οποίο μπορεί να περιέχει υπεράριθμα υπό-αντικείμενα και σετ πληροφοριών που το χαρακτηρίζουν. Επιπλέον κάθε αντικείμενο μπορεί να καλεί ένα υπεράριθμο αριθμό μεθόδων που μπορούν να δημιουργούν νέα

αντικείμενα ή να τροποποιούν το αντικείμενο αυτό και τα υπό-αντικείμενά του. Με χρήση λοιπόν αντικειμενοστραφή προγραμματισμού, ένα Inventor API μπορεί να διαχειρίζεται τα αντικείμενα και τις μεθόδους του Inventor, δηλαδή μπορεί να δημιουργεί νέα αντικείμενα σε αυτό, να τα τροποποιεί ή να αντλεί τις πληροφορίες τους. Έτσι ένα Inventor API μπορεί να αυτοματοποιεί τις όποιες διαδικασίες είναι αναγκαίο ένας χρήστης να φέρει εις πέρας.

2.4.2 Inventor VBA

Όπως αναφέρθηκε ο αντικειμενοστραφής προγραμματισμός που υλοποιήθηκε γράφτηκε σε Visual Basic και επομένως μπορούμε να χαρακτηρίσουμε ως ένα Inventor VBA την εφαρμογή που δημιουργήθηκε στο σύνολό της. Γενικότερα VBA μπορεί να θεωρηθεί ένα λογισμικό το οποίο χρησιμοποιεί τη γλώσσα προγραμματισμού Visual Basic με σκοπό την αυτοματοποίηση εργασιών εντός μίας ή περισσότερων εφαρμογών. Ένα VBA εκτελείται χρησιμοποιώντας μέρος του λογισμικού παλαιού τύπου Visual Basic της Microsoft Corporation. Να αναφερθεί ότι στην πλειοψηφία των περιπτώσεων τα VBAs λογισμικά χρησιμοποιούνται στο λειτουργικό σύστημα Windows όπου αναπτύσσονται και εκτελούνται με σκοπό την αυτοματοποίηση ενεργειών σε εφαρμογές του Microsoft Office όπως Word, Excel, PowerPoint, Access, Publisher και Visio.

Ένα λογισμικό VBA βασίζεται σε συμβάντα, και χρησιμοποιείται με σκοπό την ελεγχόμενη πυροδότηση και εκτέλεση μιας σειράς ενεργειών, από έναν χρήστη. Οι εντολές που προγραμματίζουν ένα VBA χαρακτηρίζονται ως μακροεντολές. Μια μακροεντολή είναι μια σειρά από εντολές (οδηγίες) που ομαδοποιούνται ως μια ενιαία εντολή με σκοπό την ολοκλήρωση μιας εργασίας (πολλών ενεργειών) αυτόματα. Έτσι λοιπόν με την χρήση μόνο μίας πιο σύνθετης εντολής (μακροεντολή) ολοκληρώνονται πιο σύνθετες εργασίες.

Το VBA δεν είναι ένα αυτόνομο πρόγραμμα, αλλά ένα λογισμικό που μπορεί να χρησιμοποιηθεί για τη δημιουργία συναρτήσεων που καθορίζονται από το χρήστη (User Defined Function - UDF) ή για την δημιουργία APIs όπου αυτοματοποιούν συγκεκριμένες διαδικασίες και υπολογισμούς. Δεν απαιτείται αγορά του λογισμικού VBA επειδή το λογισμικό αυτό εκτελείται με έκδοση της Visual Basic που εγκαθιστάτε αυτόματα και δωρεάν με το Microsoft Office κατά την εγκατάσταση του λογισμικού των Windows.

Ένα λογισμικό VBA όπως αναφέρθηκε έχει σκοπό να αυτοματοποιεί εργασίες εντός μίας ή περισσότερων εφαρμογών, επομένως είναι απαραίτητη η εγκατάσταση των εκάστοτε εφαρμογών προκειμένου να μπορεί το VBA να εκτελείται και να επιδρά εντός τους.

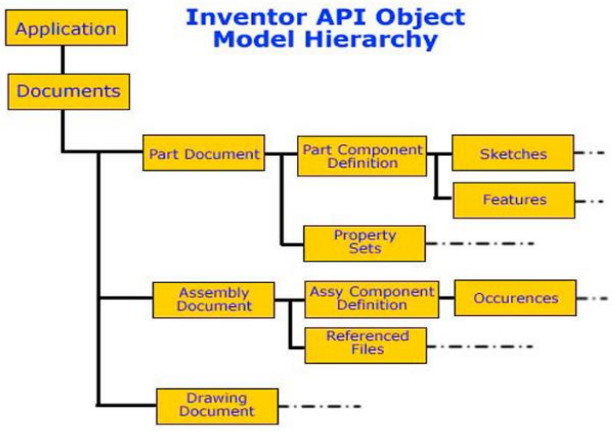
Επομένως πρέπει να είναι ξεκάθαρο ότι προκειμένου να είναι εφικτή η εκτέλεση μίας εφαρμογής Inventor VBA, είναι απαραίτητη η εγκατάσταση της εφαρμογής Inventor.

Για την υλοποίηση λοιπόν του Inventor VBA που δημιουργήθηκε, συντάχθηκε κώδικας σε Visual Basic εντός του Microsoft Visual Studio, το οποίο είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (Integrated Development Environment – IDE). Προκειμένου να είναι δυνατόν να ενεργεί ο κώδικας εντός του Autodesk Inventor χρησιμοποιήθηκε η βιβλιοθήκη autodesk.inventor.interop.dll, η οποία είναι μία βιβλιοθήκη δυναμικής σύνδεσης (Dynamic Link Library - DLL).

2.4.3 Ιεραρχία Αντικειμένων στο Inventor

Όπως αναφέρθηκε το API υλοποιήθηκε να επενεργεί με σκοπό την δημιουργία αντικείμενων εντός της εφαρμογής Autodesk Inventor και προγραμματίστηκε με χρήση αντικειμενοστραφή προγραμματισμού. Συνεπώς απαραίτητη ήταν η άριστη γνώση της ιεραρχίας των αντικειμένων που το Inventor API διαχειρίζεται, όπως και η άριστη γνώση των ενεργειών που το κάθε αντικείμενο είναι επιτρεπτό να πραγματοποιεί (μέθοδοι).

Στο εικόνα που ακολουθεί εμφανίζεται η αναπαράσταση του υψηλότερου τμήματος του μοντέλου ιεραρχίας αντικειμένων του Autodesk Inventor. Στο υψηλότερο επίπεδο όλων βρίσκεται το αντικείμενο τύπου Application το οποίο είναι το υπερ-αντικείμενο της εφαρμογής του Inventor. Όλα τα υπόλοιπα αντικείμενα είναι υπο-αντικείμενά του και επομένως είναι είτε άμεσα είτε έμμεσα συνδεδεμένα με αυτό. Το πλήρες μοντέλο ιεραρχίας αντικειμένων του Autodesk Inventor είναι υπέρογκο και έτσι θα ακολουθήσει η επεξήγηση μόνο ενός μικρού μέρους του. Στην εικόνα που ακολουθεί αποφαίνεται η κορυφή του μοντέλου ιεραρχίας αντικειμένων του Autodesk Inventor.



ΣΧΗΜΑ 2.12 – ΚΟΡΥΦΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΙΕΡΑΡΧΙΑΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΤΟΥ AUTODESK INVENTOR

Στο σημείο αυτό η χρησιμότητά της βιβλιοθήκης δυναμικής σύνδεσης (DLL) "autodesk.inventor.interop.dll" πρέπει να γίνει σαφής, αφού αυτή συνδέει δυναμικά τις μεταβλητές του κώδικα με όλες τις απαραίτητες πληροφορίες που φέρει ένα αντικείμενο του Inventor. Έτσι σε επίπεδο προγραμματισμού επιτρέπει την δήλωση μεταβλητών ως αντικείμενα του Inventor. Συνεπώς οποιαδήποτε μεταβλητή δηλώνεται με τύπο όπου ξεκινάει με το πρόθεμα "Inventor.", θεωρείται αντικείμενο του Inventor διαμέσου τις βιβλιοθήκης αυτής. Ο κώδικας που ακολουθεί παρακάτω είναι γραμμένος στη γλώσσα προγραμματισμού Visual Basic και επιτυγχάνει τη σύνδεση του API με την εφαρμογή του Inventor. Στο παράδειγμα αυτό δηλώνεται η μεταβλητή με όνομα "myApplication" ως ένα αντικείμενο τύπου Inventor.Application, το οποίο πρακτικά είναι ολόκληρη την εφαρμογή.

```
Dim myApplication As Inventor.Application
myApplication = GetObject(, "Inventor.Application")
```

Πλέον όλες οι απαραίτητες πληροφορίες όλων τον αντικειμένων του Inventor αντιπροσωπεύεται από τη μεταβλητή "myApplication" η οποία είναι δηλωμένη ως ένα αντικείμενο τύπου Inventor.Application το οποίο είναι το υπερ-αντικείμενο όλων τον αντικειμένων του Inventor. Όλα τα δεδομένα λοιπόν του Inventor CAD μπορούν να προσπελαστούν μέσω του δηλωμένου αντικειμένου "myApplication. Μέσω αυτού του αντικειμένου συνεπώς είναι δυνατόν να δρομολογηθεί μια οποιαδήποτε ενέργεια εντός του Inventor η οποία θα μπορεί να δημιουργεί νέα αντικείμενα, να τροποποιεί υπάρχοντα αντικείμενα, ή να αντλεί πληροφορίες από αυτά. Επιπλέον αφού το αντικείμενο αυτό είναι υπερ-αντικείμενο όλων, μπορεί να περιέχει ως υπο-αντικείμενα οτιδήποτε δημιουργείται εντός των περιβάλλοντων Part, Assembly, Drawing και Presentation. Συνεπώς ένα τέτοιο αντικείμενο τύπου Inventor.Application, μπορεί να χρησιμοποιείται για την δημιουργία οποιουδήποτε αντικειμένου εντός οποιοδήποτε περιβάλλοντος του Inventor.

Γενικεύοντας λοιπόν μπορούμε να πούμε ότι, μία οποιαδήποτε μεταβλητή δηλωμένη ως αντικείμενο τύπου Inventor.Application, μπορεί να χρησιμοποιείται ως πύλη για την δημιουργία και τροποποίηση νέων υπο-αντικειμένων, την άντληση των πληροφοριών τους ή την μεταφορά τους από ένα περιβάλλον του Inventor σε ένα άλλο.

Στο παράδειγμα που ακολουθεί ο κώδικας αυτός χρησιμοποιεί την μεταβλητή "myApplication" τύπου Inventor.Application που δηλώθηκε παραπάνω. Αυτή η μεταβλητή χρησιμοποιείται ως πύλη για την σύνδεση νέων μεταβλητών με αντικείμενα του Inventor.

```
Dim partDoc As Inventor.PartDocument
Dim partComp As Inventor.SheetMetalComponentDefinition
partDoc = myApplication.ActiveDocument
partDoc.SubType = "{9C464203-9BAE-11D3-8BAD-0060B0CE6BB4}"
partComp = partDoc.ComponentDefinition
```

Σύμφωνα λοιπόν με το ανωτέρω παράδειγμα, στις πρώτες δύο εντολές αρχικά δηλώνονται οι μεταβλητές partDoc και partComp ως αντικείμενα τύπου Inventor.PartDocument και Inventor.SheetMetalComponentDefinition αντίστοιχα.

Σύμφωνα με το μοντέλο ιεραρχίας των αντικειμένων του Inventor, ένα αντικείμενο τύπου Inventor.PartDocument πρέπει να είναι υπο-αντικείμενο του υπερ-αντικειμένου Inventor.Application. Έτσι στην 3η εντολή του παραδείγματος, η μεταβλητή "partDoc" συνδέεται ως υπο-αντικείμενο του αντικειμένου "myApplication". **Ουσιαστικά με την εκτέλεση της 3ης εντολής, πυροδοτούνται οι κατάλληλες ενέργειες ώστε εντός του υπερ-αντικειμένου** "myApplication" **να δημιουργηθεί ένα ενεργό αρχείο τύπου Part το οποίο θα ακούει πλέον στο όνομα** "partDoc".

Πλέον στην 4η εντολή η μεταβλητή "partDoc" είναι ένα αντικείμενο τύπου **Inventor.** PartDocument του οποίου δηλώνεται ο δευτερεύοντας τύπος (Sheet Metal).

Έπειτα στην 5η εντολή η μεταβλητή "partComp" ως ένα αντικείμενο τύπου Inventor.SheetMetalComponentDefinition, πρέπει να συνδεθεί ως υπο-αντικείμενο με ένα αντικειμένου τύπου Inventor.PartDocument. **Έτσι στην 5η εντολή η μεταβλητή** "partComp" **συνδέεται ως υπο-αντικείμενο της μεταβλητής** "partDoc"**, και πλέον στο Part περιβάλλον του Inventor προσδίδονται μερικές ακόμα δυνατότητες που επιτρέπουν την ανάπτυξη ενός τρισδιάστατου μοντέλου (περιβάλλον Sheet Metal Part). Επομένως πλέον έχει δημιουργηθεί ένα περιβάλλον** Sheet Metal Part **το οποίο ακούει στο όνομα** "partComp". Επομένως όλες οι ενεργείς στο εξής θα πρέπει να γίνουν στο περιβάλλον Sheet Metal Part θα έχουν ως βάση την μεταβλητή "partComp".

Προφανώς ακόμα και στην 5η εντολή , πρόδομος για την δημιουργία του αντικειμένου "partComp" είναι το "myApplication" πέραν του "partDoc", αφού αν γίνει η αντικατάστασή του θα προκύψει η μακροεντολή:

```
partComp = myApplication.ActiveDocument.ComponentDefinition
```

Έχοντας υλοποιήσει λοιπόν τα προαναφερθέντα, ο κώδικας που ακολουθεί στο παρακάτω παράδειγμα δημιουργεί ένα Sketch εντός του περιβάλλοντος Sheet Metal Part. Το Sketch αυτό υλοποιείτε να περιέχει έναν κύκλο με διάμετρο (Diam) και κέντρο το (0,0).

```
Dim mySketch As Inventor.PlanarSketch
mySketch = partComp.Sketches.Add(partComp.WorkPlanes.Item(2))
mySketch.SketchCircles.AddByCenterRadius(myApplication.oTG.CreatePoint2d(0,0), Diam)
```

Στην 1η εντολή του ανωτέρω κώδικα δηλώνεται ένα αντικείμενο τύπου Inventor.PlanarSketch το οποίο θα ακούει πλέον στο όνομα "mySketch".

Στην 2η εντολή μπορεί να γίνει φανερό πώς το αντικείμενο "mySketch", συνδέεται ως υπο-αντικείμενο του "partComp" του προηγούμενου παραδείγματος. Έτσι με το πέρας της εντολής αυτής ένα δισδιάστατο Sketch θα έχει δημιουργηθεί και τοποθετηθεί εντός του δεύτερου επιπέδου (WorkPlane) που το "partComp" περιέχει. Τοποθετείται δηλαδή εντός ενός εκ των τριών προκαθορισμένων επιπέδων (Plane) που περιέχονται εντός του τρισδιάστατου εικονικού περιβάλλοντος του Sheet Metal Part.

Με την 3η εντολή, στο εν λόγω δισδιάστατο Sketch, με χρήση κατάλληλης μεθόδου τοποθετείται ένας κύκλος με διάμετρο (Diam) και κέντρο το (0,0).

Γενικότερα, με σκοπό την δημιουργία ενός τρισδιάστατου μοντέλου, ένα Sketch θα πρέπει να περιέχει τουλάχιστον μία κλειστή επιφάνια (Inventor.Profile) με την χρήση της οποίας θα μπορεί να δοθεί στο Sketch διάσταση στον χώρο. Έτσι αφού το "mySketch" που δημιουργήθηκε περιέχει έναν κύκλο, με κατάλληλες μεθόδους είναι δυνατόν αυτό να δημιουργήσει (προς δύο κατευθύνσεις) έναν επιφανειακό ή συμπαγή κύλινδρος.

Με παρόμοια λογική λοιπόν μπορεί να επιτυγχάνεται ο προγραμματισμός όλων των απαραίτητων ενεργειών τόσο εντός του περιβάλλοντος Part όσο και εντός του περιβάλλοντος Drawing. Επομένως έχοντας ως σκοπό την αυτοματοποιημένη παραγωγή ενός Μηχανολογικού Σχεδίου, οποιαδήποτε ενέργεια εντός του Inventor μπορεί να εκτελεστεί από το API με τη χρήση του αντικειμενοστραφή προγραμματισμού που περιγράφηκε. Δεν υπάρχει λόγος για περεταίρω επεξήγηση καθώς το πλήρες μοντέλο ιεραρχίας αντικειμένων του Autodesk Inventor είναι χαοτικά υπέρογκο. Ομοίως χαοτικά υπέρογκη είναι και η θεωρεία που διέπει τις μεθόδους που το κάθε ένα αντικείμενο είναι δυνατόν να καλεί. Το συνολικό διάγραμμα του μοντέλου ιεραρχίας αντικειμένων του Autodesk Inventor είναι τόσο μεγάλο που δεν είναι δυνατόν να παρουσιαστεί ολόκληρο, επομένως θα είναι

βολικότερη η ανασκόπησή του, διαμέσου της αναζήτησής του στο διαδίκτυο χρησιμοποιώντας τον επακόλουθο ιστότοπο: https://knowledge.autodesk.com/akn-aknsite-article-attachments/d20aa033-13a7-4b23-a790-1897b317c523.pdf.

Πριν κλείσει το παρόν υποκεφάλαιο να αναφερθεί ότι το Microsoft Visual Studio, με την προσθήκη της DLL βιβλιοθήκης "autodesk.inventor.interop.dll", μπορεί να αναγνωρίζει κατά την δήλωση μεταβλητών ποια αντικείμενα αυτή διαχειρίζεται, δίχως να είναι απαραίτητη η χρήση του προθέματος "Inventor.". Επομένως ο κώδικας των παραπάνω παραδειγμάτων μπορεί συνολικά να γραφεί πιο απλά σύμφωνα με τον παρακάτω.

```
Dim myApplication As Application
myApplication = GetObject(, "Inventor.Application")
Dim partDoc As PartDocument
Dim partComp As SheetMetalComponentDefinition
partDoc = myApplication.ActiveDocument
partDoc.SubType = "{9C464203-9BAE-11D3-8BAD-0060B0CE6BB4}"
partComp = partDoc.ComponentDefinition
Dim mySketch As PlanarSketch
mySketch = partComp.Sketches.Add(partComp.WorkPlanes.Item(2))
mySketch.SketchCircles.AddByCenterRadius(myApplication.oTG.CreatePoint2d(0,0), Diam)
```

2.5 Ανάπτυξη του GUI μέσω των Windows Forms

Τα Windows Forms είναι ένα σύνολο βιβλιοθηκών που εξυπηρετούν στην ανάπτυξη εμπλουτισμένων γραφικών περιβάλλοντων σε εφαρμογές των Windows. Είναι ένα γραφικό API για την εμφάνιση δεδομένων και τη διαχείριση των αλληλεπιδράσεων των χρηστών. Τα Windows Forms είναι σχεδιασμένα με αρχιτεκτονική παρόμοια με αυτή των προγραμμάτων των Windows, η οποία βασίζεται σε συμβάντα (clicks ή enters) εντός γραφικών στοιχείων.

Οι υλοποιήσεις με Windows Forms εντός του Microsoft Visual Studio είναι πολύ βολικές για τον προγραμματιστή καθώς ένας οπτικός σχεδιαστής παρέχεται ο οποίος επιτρέπει την τοποθέτηση των γραφικών στοιχείων εντός του με χρήση μεταφοράς και απώθησης (drag and drop).

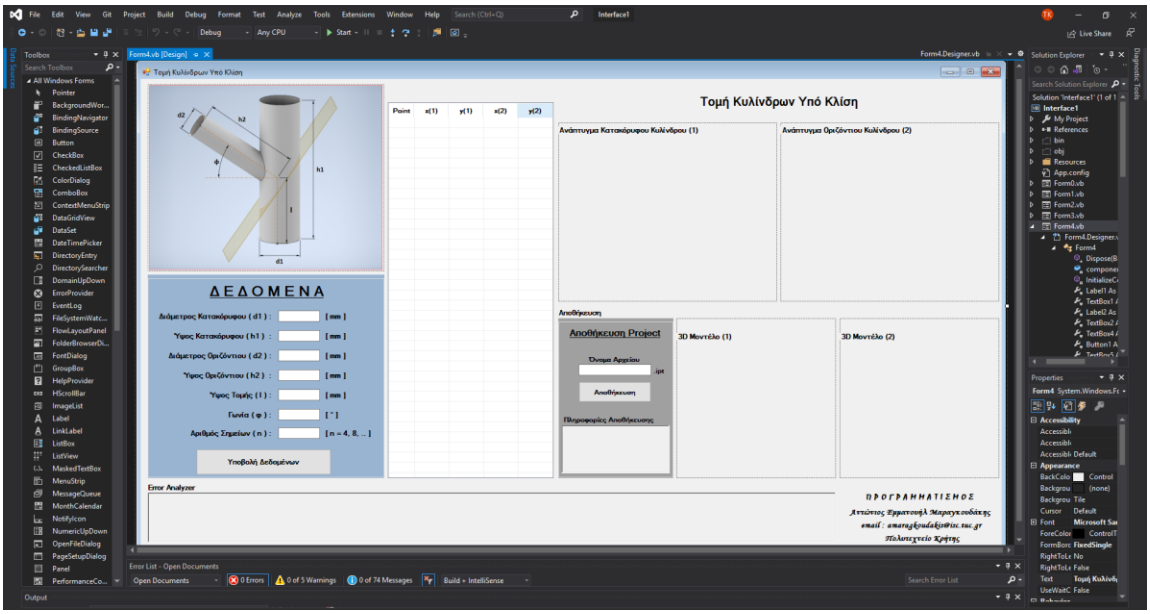
Έπειτα από την απώθηση (drop) ενός γραφικού στοιχείου σε ένα σημείο του οπτικού σχεδιαστή, αυτόματα δημιουργείται μέρος του κώδικα που διέπει την λειτουργεία του γραφικού στοιχείου αλλά και την τοποθεσία του. Γενικότερα, υπεράριθμα γραφικά στοιχεία με πολυπληθείς διαφορετικές ιδιότητες υποστηρίζονται από τα Windows Forms όπως: Textboxes, ListBoxes, ListViews, Buttons, MessageBoxes, Labels και πολλά άλλα.

Στην παρούσα διπλωματική λοιπόν, με την χρήση των Windows Forms υλοποιήθηκε εντός του Visual Studio μία γραφική διεπαφή χρήστη (GUI) η οποία έχει σαν σκοπό να δέχεται δεδομένα από έναν χρήστη και να επιστρέφει σε αυτόν πληροφορίες. Επιπλέον υλοποιήθηκε όταν ο χρήστης επιθυμεί, να έχει την δυνατότητα να αποθηκεύσει τα παραγόμενα του API (3D μοντέλα .ipt, υπολογισμοί .txt και Μηχανολογικό Σχέδιο .dwg και .pdf). Τα δεδομένα που πρέπει να δίνει ο χρήστης δεν είναι άλλα από τις διαστάσεις που περιγράφουν το εκάστοτε τρισδιάστατο μοντέλο Κυλινδρικών Τομών.

Με την ανίχνευση λοιπόν ενός συμβάντος (click – enter) σε ένα γραφικό στοιχείο πραγματοποιούνται οι αντίστοιχες ενέργειες που ο χρήστης ζητά. Στην περίπτωσή μας αυτό σημαίνει ότι με την Υποβολή των δεδομένων από τον χρήστη, ενεργοποιείται η εκτέλεση των εντολών του Inventor API που περιγράφηκαν στο προηγούμενο υποκεφάλαιο. Έπειτα από την ολοκλήρωση της εκτέλεσης των εντολών αυτών, αν ανιχνευτεί αλληλεπίδραση του χρήστη με το γραφικό στοιχείο της Αποθήκευσης, τότε ενεργοποιείται ο κώδικας που πραγματοποιεί την αποθήκευση των αρχείων που το API παρήγαγε (.ipt, .dwg, .pdf, .txt). Τέλος να αναφερθεί ότι εκτέλεση των εντολών, δηλαδή οι ενέργειες που πραγματοποιούνται εντός του Inventor, γίνονται (κρυφά) στο background, δηλαδή πίσω από την γραφική διεπαφή που ο χρήστης βλέπει.

Γενικότερα, πολλά διαφορετικά παράθυρα γραφικών διεπαφών μπορούν να δημιουργηθούν και να περιέχονται στην ίδια εφαρμογή, τα οποία σε επίπεδο προγραμματισμού ονομάζονται φόρμες (Forms). Έτσι ανάλογα με τα γεγονότα που ανιχνεύονται από τον χρήστη, διαφορετικές φόρμες μπορούν να προβάλλονται σε αυτόν. Η δυνατότητα αυτή χρησιμοποιείται στην εφαρμογή που αναπτύχτηκε, προκειμένου ο χρήστης να μπορεί να επιλέξει από την αρχική φόρμα (μενού), την επόμενη φόρμα (υπολογισμού), μέσω της οποίας θα χειρίζεται την περίπτωση για την οποία επιθυμεί να παραχθεί το Μηχανολογικό Σχέδιο. Έπειτα από την επιλογή του λοιπόν μία νέα φόρμα θα παρουσιάζεται σε αυτόν, διαφορετική για την κάθε μία από τις πέντε περιπτώσεις Κυλινδρικών Τομών.

Στην εικόνα που ακολουθεί παρουσιάζεται ως παράδειγμα ένα παράθυρο γραφικής διεπαφής χρήστη (Form), που δημιουργήθηκε για την 4η περίπτωση Κυλινδρικών Τόμων. Η εικόνα αυτή δείχνει το εν λόγω παράθυρο, όπως αυτό φαίνεται μέσα από τον οπτικό σχεδιαστή του Visual Studio, δηλαδή κατά την διαδικασία σχεδίασης και προγραμματισμού.



ΣΧΗΜΑ 2.13 – ΥΛΟΠΟΙΗΣΗ ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ ΧΡΗΣΤΗ ΕΝΤΟΣ ΤΟΥ VISUAL STUDIO

Όλες αυτές οι φόρμες υπολογισμού μοιάζουν αρκετά μεταξύ τους και είναι παρόμοιες με αυτήν της εικόνας που παρουσιάστηκε. Αυτό αφού η εν λόγω φόρμα, όπως και οι υπόλοιπες τέσσερεις, έχουν κοινό σκοπό. Κοινός σκοπός τους είναι η παροχή δυνατοτήτων στον χρήστη που του επιτρέπουν να υποβάλει δεδομένα, να έχει πρόσβαση σε πληροφορίες που του παρουσιάζονται και να αποθηκεύει τα παραγόμενα αρχεία αν επιθυμεί.

Με τον συνδυασμό λοιπόν των Windows Forms και του Inventor API μία ολοκληρωμένη εφαρμογή υλοποιήθηκε, η οποία στο background μπορεί να εκτελεί ενέργειες εντός του Inventor, ενώ παράλληλα μπορεί να παρέχει σε έναν χρήστη δυνατότητες αλληλεπίδρασης όπως: είσοδος, διάγνωση σφαλμάτων, έλεγχος αποτελεσμάτων και αποθήκευση.

Κεφάλαιο 3

Προσέγγιση Και Υλοποίηση

Στο εν λόγω κεφάλαιο πρόκειται να περιγραφούν όλα τα βήματα που ακολουθήθηκαν στον βωμό της υλοποίησης αυτής της διπλωματικής εργασίας. Θα γίνει δηλαδή πλήρης περιγραφή όλων απαραίτητων βημάτων, από την εγκατάσταση όλων των απαραίτητων εργαλείων (Autodesk Inventor, Microsoft Visual Studio), μέχρι και την κατάληξη του τελικά ανεπτυγμένου λογισμικού Inventor VBA που δημιουργήθηκε.

Πιο συγκεκριμένα στο κεφάλαιο αυτό θα επεκταθούμε κυρίως στα κάτωθι:

- Θα περιγραφεί το πρώιμο στάδιο στο οποίο έγινε η εγκατάσταση των απαραίτητων εργαλείων, καθώς και η κατανοητή των λειτουργειών τους.

- Θα περιγραφεί αναλυτικά η διαδικασία ανάπτυξης της γραφικής διεπαφής χρήστη (GUI) καθώς και ο τρόπος επικοινωνίας αυτού με το Inventor API.

- Θα περιγραφούν αναλυτικά όλες οι ενέργειες που το API εκτελεί, οι λόγοι που αυτές λαμβάνουν τόπο καθώς και τα αποτελέσματα που αυτές επιφέρουν εντός της εφαρμογής του Inventor.

- Θα γίνει η γεωμετρική περιγραφή που επεξηγεί τις συνθήκες ελέγχου των αποδεκτών τιμών εισόδου σε όλες τις περιπτώσεις Κυλινδρικών Τομών.

3.1 Πρώιμο Στάδιο

Η εγκατάσταση του Autodesk Inventor ήταν η πρώτη ενέργεια που πραγματοποιήθηκε. Αυτό φυσικά αφού η εφαρμογή αυτή είναι απαραίτητη γιατί το API θα πρέπει να πραγματοποίει ενέργειες εντός της. Ένας χρήστης δεν μπορεί να έχει πρόσβαση στο περιβάλλον του Inventor όσο το API εκτελεί ενέργειες σε αυτό, αλλά μόνο μπορεί να αλληλοεπιδρά με την γραφική διεπαφή GUI που δημιουργήθηκε. Το GUI δηλαδή λειτουργεί σαν μία μάσκα που κρύβει από τον χρήστη την ύπαρξη του Inventor, αλλά αυτό δεν καταργεί την απαραίτητη λειτουργεία του στο background, και επομένως η εγκατάσταση του Inventor είναι απαραίτητη σε κάθε περίπτωση.

Δεύτερη ενέργεια όπως είναι προφανές ήταν ή εγκατάσταση του δεύτερου απαραίτητου εργαλείου, το οποίο δεν είναι άλλο από το Visual Studio, το οποίο είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (IDE) της Microsoft. Όπως έχουμε αναφέρει το Visual Studio είναι απαραίτητο τόσο για την ανάπτυξη μίας γραφικής διεπαφής χρήστη (GUI) με χρήση των βιβλιοθηκών των Widows Forms, όσο και για την ανάπτυξη του Inventor API που θα εκτελεί ενέργειες εντός του Inventor χρησιμοποιώντας την βιβλιοθήκη "autodesk.inventor.interop.dll".

Σε επόμενο επίπεδο λοιπόν έπρεπε να γίνει γενικότερα κατανοητή η λειτουργεία των δύο εργαλείων που είχαν εγκατασταθεί.

3.1.1 Κατανόηση του Autodesk Inventor

Απαραίτητη λοιπόν ήταν η εκμάθησης της CAD εφαρμογής Autodesk Inventor, σε επίπεδο χρήστη. Αυτό αφού έπρεπε να γίνει ξεκάθαρα κατανοητή η ακολουθία ενεργειών όπου το API έπρεπε να φέρει εις πέρας για κάθε περίπτωση Κυλινδρικών Τομών. Έτσι η οποιαδήποτε ενέργεια θα ήταν απαραίτητο να εκτελείτε από το API, πρώτα σε επίπεδο χρήστη έγινε αντιληπτή η ίδια, όπως και οι προ-απαιτούμενες ενέργειές της, αλλά και οι γεωμετρικοί περιορισμοί της. Να γίνει σαφές το ότι η εκμάθηση της χρήσης του Inventor αφορούσε μόνο τα περιβάλλοντα Part και Drawing όπου ήταν απαραίτητα για την υλοποίηση της εργασίας.

Αρχικά λοιπόν σε επίπεδο χρήστη, υλοποιήθηκαν στο περιβάλλον Part η κατασκευή των Τρισδιάστατων Μοντέλων, η εκπλήρωση των κανόνων ανάπτυξης καθώς και η ανάπτυξη. Έτσι έγινε αντιληπτή η βασική ακολουθία των ενεργειών που το API αποφασίστηκε να εκτελεί εντός του Part περιβάλλοντος.

Στην συνέχεια πάλι σε επίπεδο χρήστη, η ίδια διαδικασία έλαβε τόπο και στο Drawing περιβάλλον. Δηλαδή υλοποιήθηκαν η επιλογή διαστάσεων του φύλλου σχεδιάσεως, η τοποθέτηση των δισδιάστατων όψεων των αναπτυγμάτων, η τοποθέτηση όψεων των τρισδιάστατων αντικειμένων, η τοποθέτηση διαστάσεων μήκους, η δημιουργία πίνακα τιμών, η δημιουργία υπομνήματος και άλλα. Παρομοίως λοιπόν έγινε αντιληπτή και η βασική ακολουθία των ενεργειών που το API αποφασίστηκε να εκτελεί στο Drawing περιβάλλον, οι οποίες είναι υπεύθυνες για την παραγωγή του Μηχανολογικού Σχεδίου.

3.1.2 Κατανόηση του Microsoft Visual Studio

Απαραίτητη ήταν επίσης η κατανόηση της λειτουργείας του Microsoft Visual Studio. Έτσι έπειτα την εγκατάστασή του, έγινε η σύνδεση αυτού με τις απαραίτητες βιβλιοθήκες των Windows Forms όπως και με την βιβλιοθήκη δυναμικής σύνδεσης autodesk.inventor.interop.dll. Έπειτα ακολούθησε η γραφή και εκτέλεση πειραματικών μακροεντολών με σκοπό τόσο την κατανόηση της Visual Basic όσο και των παροχών που η πλατφόρμα Visual Studio διαθέτει.

Στο επίπεδο λοιπόν των Windows Forms υλοποιήθηκε αρχικά η ανάπτυξη μίας απλής γραφικής διεπαφής χρήστη, με σκοπό την εκμάθηση των μορφοποιήσεων των διαθέσιμων γραφικών αντικειμένων (ListBox, ListView, Label, etc.). Έπειτα υλοποιήθηκαν εντολές που διαχειρίζονται αυτά τα γραφικά αντικείμενα, με σκοπό την κατανόηση των δυνατοτήτων που αυτά παρέχουν σε επίπεδο κώδικα. Έτσι τελικά επιτεύχθηκε ο πρώτος στόχος σε επίπεδο γραφικών στοιχείων που δεν ήταν άλλος από την τοποθέτησή τους, την ανίχνευση γεγονότων (click ή enter) εντός αυτών και την άντληση ή παρουσίαση πληροφοριών από ή προς αυτά.

Στο επίπεδο του Inventor API, παρομοίως έγιναν εκτελέσεις διάφορων εντολών που η βιβλιοθήκη autodesk.inventor.interop.dll μας επιτρέπει να εκτελούμε. Έτσι για αρχή ανεξάρτητα από το GUI έγιναν κατανοητές οι εντολές που ενεργούν εντός του Inventor. Οι εντολές που έχουν περιγραφεί στο υπο-κεφάλαιο 2.4.3 ήταν οι πρώτες που έγιναν κατανοητές σε πρακτικό επίπεδο και έπειτα πάρα πολλές ακόμη ακολούθησαν.

Αφού λοιπόν είχε γίνει κατανοητός ο προγραμματισμός τόσο με χρήση των βιβλιοθηκών των Windows Forms όσο και με χρήση της βιβλιοθήκης autodesk.inventor.interop.dll, επόμενο βήμα ήταν η γραφή του κώδικα που διέπει την εφαρμογή Inventor VBA που τελικά δημιουργήθηκε. Στα επόμενα υπο-κεφάλαια λοιπόν ακολουθεί η περιγραφή και επεξήγηση της υλοποιημένης εφαρμογής.

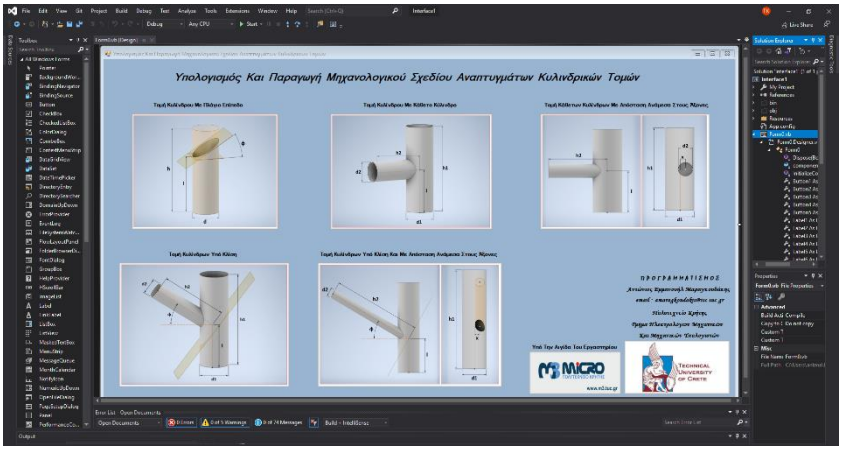
3.2 Η Ανάπτυξη του GUI και η Σύνδεσή του με το Inventor API

Σε πρώτο επίπεδο έπρεπε να αναπτυχθεί ένα λειτουργικό GUI το οποίο έπειτα θα έπρεπε να ενσωματώσει εντός του το Inventor API όπως θα περιγραφεί σε αυτό το υποκεφάλαιο. Θα περιγραφεί λοιπόν η λειτουργικότητα που αποφασίστηκε να παρέχει στον χρήστη το GUI που αναπτύχθηκε, καθώς και η διαδικασία ανάπτυξής του. Τέλος θα ακολουθήσει η περιγραφή της σύνδεσής του με το Inventor API.

3.2.1 Ανάπτυξη της Γραφικής Διεπαφής Χρήστη (GUI)

Όπως έχει αναφερθεί, μια γραφική διεπαφή χρήστη μπορεί να αποτελείτε από πολλά διαφορετικά παράθυρα (Forms), τα οποία περιέχουν γραφικά στοιχεία. Τα εν λόγω παράθυρα δεν παρουσιάζονται όλα μαζί ταυτόχρονα στον χρήστη, αλλά προβάλλονται επιλεκτικά σε αυτόν σύμφωνα με τις ενέργειες του.

Η φόρμα που αρχικά παρουσιάζεται στον χρήστη κατά την εκκίνηση της εφαρμογής, υλοποιήθηκε να περιέχει πέντε ίδια γραφικά στοιχεία τύπου Button, ένα για κάθε μία από τις πέντε περιπτώσεις Κυλινδρικών Τομών όπου και αυτοματοποιούνται. Κατά την ανίχνευση ενός γεγονότος (click ή enter) σε ένα τέτοιο γραφικό στοιχείο (Button), κατάλληλες ενέργειες εμφανίζουν στον χρήστη μία νέα φόρμα (υπολογισμού), η οποία ταυτόχρονα όσο παραμένει ενεργή θα κρύβει και την ύπαρξη της αρχικής. Στην εικόνα που ακολουθεί λοιπόν αποφαίνεται η εν λόγω αρχική φόρμα (μενού), εντός του οπτικού σχεδιαστή του Visual Studio, κατά την διαδικασία του προγραμματισμού της.



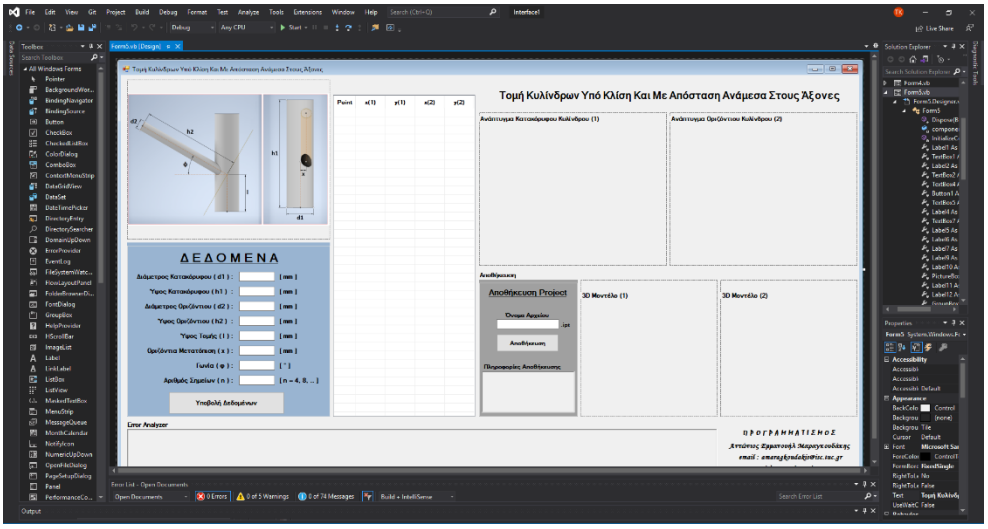
ΣΧΗΜΑ 3.1 – ΑΡΧΙΚΗ ΓΡΑΦΙΚΗ ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ ΕΝΤΟΣ ΤΟΥ VISUAL

Ομοίως με το Inventor API, ο κώδικας που χειρίζεται τα γραφικά στοιχεία των Windows Form, διέπετε από αντικειμενοστραφή προγραμματισμό. Αυτό αφού κάθε φόρμα και κάθε γραφικό στοιχείο που αυτή περιέχει θεωρούνται αντικείμενα σε επίπεδο προγραμματισμού. Στην συνέχεια ακολουθεί και περιγράφεται κώδικας όπου διαχειρίζεται ένα γεγονός στο 5ο Button της φόρμας που παρουσιάστηκε παραπάνω, το οποίο Button πρακτικά είναι η 5η εικόνα Κυλινδρικών Τομών που περιέχεται.

```
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
    Dim myForm As New Form5
    myForm.ShowDialog()
End Sub
```

Ο ανωτέρω κώδικας λοιπόν, με την ανίχνευση ενός click ή enter στο εν λόγω Button, δημιουργεί ένα αντικείμενο τύπου Form5 το οποίο πλέον θα ακούει στο όνομα "**myForm**". Το αντικείμενο αυτό είναι μία φόρμα που έχει υλοποιηθεί και περιγράφεται παρακάτω. Εν συνέχεια με την χρήση της μεθόδου **ShowDialog()**, η εν λόγω φόρμα παρουσιάζεται στον χρήστη ενώ ταυτόχρονα η ίδια κρύβει και την ύπαρξη της αρχικής φόρμας (καθώς και κάθε άλλης πιθανόν ενεργής φόρμας γενικότερα).

Στο συγκεκριμένο παράδειγμα το αντικείμενο Form5 δεν είναι παρά μία νέα φόρμα που έχει υλοποιηθεί να επιτρέπει στον χρήστη να χειρίζεται την 5η περίπτωση Κυλινδρικών Τομών. Η εν λόγω φόρμα παρουσιάζεται στην εικόνα που ακολουθεί, ενώ τα γραφικά στοιχεία (αντικείμενα) που αυτή περιέχει περιγράφονται παρακάτω. Στην εικόνα που ακολουθεί λοιπόν αποφαίνεται η εν λόγω φόρμα, εντός του οπτικού σχεδιαστή του Visual Studio, κατά την σχεδίαση και τον προγραμματισμό της.



ΣΧΗΜΑ 3.2 – ΦΟΡΜΑ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ 5ΗΣ ΠΕΡΙΠΤΩΣΗΣ ΕΝΤΟΣ ΤΟΥ VISUAL STUDIO

Τα γραφικά στοιχεία λοιπόν που περιέχονται στην φόρμα που παρουσιάστηκε εξυπηρετούν συγκεκριμένους σκοπούς σύμφωνα με τους οποίους και αυτά περιγράφονται στην συνέχεια. Πιο συγκεκριμένα εξυπηρετούν την εισαγωγή και τον έλεγχο των δεδομένων, την παρουσίαση των αποτελεσμάτων καθώς και την αποθήκευση των παραγομένων.

Εισαγωγή και Έλεγχος των Δεδομένων

Όπως μπορεί να φανεί στην φόρμα που παρουσιάστηκε παραπάνω, ένα PictureBox είναι τοποθετημένο στην πάνω αριστερή γωνία και προβάλει μία προκαθορισμένη εικόνα που παρουσιάζει τις διαστάσεις που το τρισδιάστατο μοντέλο περιέχει με σκοπό την κατανόηση των δεδομένων που απαιτούνται ως είσοδος.

Ένα GroupBox (με γαλάζιο χρώμα) είναι τοποθετημένο από κάτω το οποίο ομαδοποιεί τα γραφικά στοιχεία που είναι υπεύθυνά για την υποβολή των δεδομένων. Ως δεδομένα υποβάλλονται οι διαστάσεις του επιθυμητού τρισδιάστατου μοντέλου καθώς και ο επιθυμητός αριθμός υπολογισμών των σημείων. Τα δεδομένα μπορούν να πληκτρολογούνται από τον χρήστη εντός των αντικειμένων TextBox που περιέχονται στο εν λόγω GroupBox. Τέλος αντικείμενα κειμένου (Label) εμπεριέχονται με σκοπό την περιγραφή των δεδομένων, ενώ ένα Button εμπεριέχεται με σκοπό την Υποβολή των δεδομένων και κατά συνέπεια την επικείμενη εκκίνηση της εκτέλεσης του Inventor API όπως θα περιγραφεί αργότερα.

Στην πιο χαμηλή περιοχή της φόρμας ένα στενόμακρο RichTextBox είναι τοποθετημένο, το οποίο έχει ως σκοπό την ενημέρωση του χρήστη με κατάλληλα διαγνωστικά μηνύματα, σε περίπτωση που αυτός έχει υποβάλει μη έγκυρες τιμές ως δεδομένα εισόδου (Error Analyzer). Οι συνθήκες που ελέγχουν την εγκυρότητα των δεδομένων θα περιγραφούν για κάθε περίπτωση Κυλινδρικών Τομών σε επόμενο υποκεφάλαιο. Εφόσον τα δεδομένα δεν είναι έγκυρα το Inventor API τελικά δεν θα εκτελείται, μέχρις εκ νέου ορθής υποβολής δεδομένων.

Παρουσίαση των Αποτελεσμάτων

Εφόσον το API έχει αρχίσει να εκτελείται, οι πληροφορίες που αυτό δημιουργεί παρουσιάζονται στο χρήστη εντός των γραφικών στοιχείων που περιγράφονται εν συνέχεια. Ένα ListView είναι τοποθετημένο λίγο πιο αριστερά από το κέντρο. Εντός του εν λόγω ListBox παρουσιάζονται οι υπολογισμένες τιμές του κάθε αναπτύγματος

έπειτα από την κάθε εκτέλεση του API. Πέντε στήλες περιέχονται σε αυτό. Η πρώτη στήλη θα απαριθμεί τα υπολογισμένα σημεία. Οι επόμενες δύο στήλες θα περιέχουν τους υπολογισμούς των x και των y τιμών των σημείων του αναπτύγματος του κάθετου κυλίνδρου, ενώ οι δύο τελευταίες στήλες περιέχουν ομοίως τους υπολογισμούς των x και των y τιμών των σημείων του αναπτύγματος του υπό γωνία κυλίνδρου.

Τέσσερα ακόμα PictureBox είναι τοποθετημένα στα δεξιά τα οποία θα παρουσιάζουν εικόνες οι οποίες θα είναι στιγμιότυπα (screenshots) που το API θα τραβάει εντός του Inventor. Τα στιγμιότυπα αυτά θα περιέχουν τα τρισδιάστατα μοντέλα και τα αναπτύγματα που θα έχουν δημιουργηθεί από το API, τόσο για τον κατακόρυφο όσο και για τον υπο γωνία κύλινδρο. Σκοπός των εν λόγω PictureBox είναι να μπορεί να οπτικοποιεί ο χρήστης τα αντικείμενα που το API θα έχει δημιουργήσει σύμφωνα με τα δεδομένα που ίδιος θα έχει υποβάλει. Έτσι ένας υποτυπώδης έλεγχος μπορεί να γίνει πριν από την αποθήκευση των παραγόμενων αρχείων (.ipt, .dwg) του Inventor.

Αποθήκευση των Παραγομένων

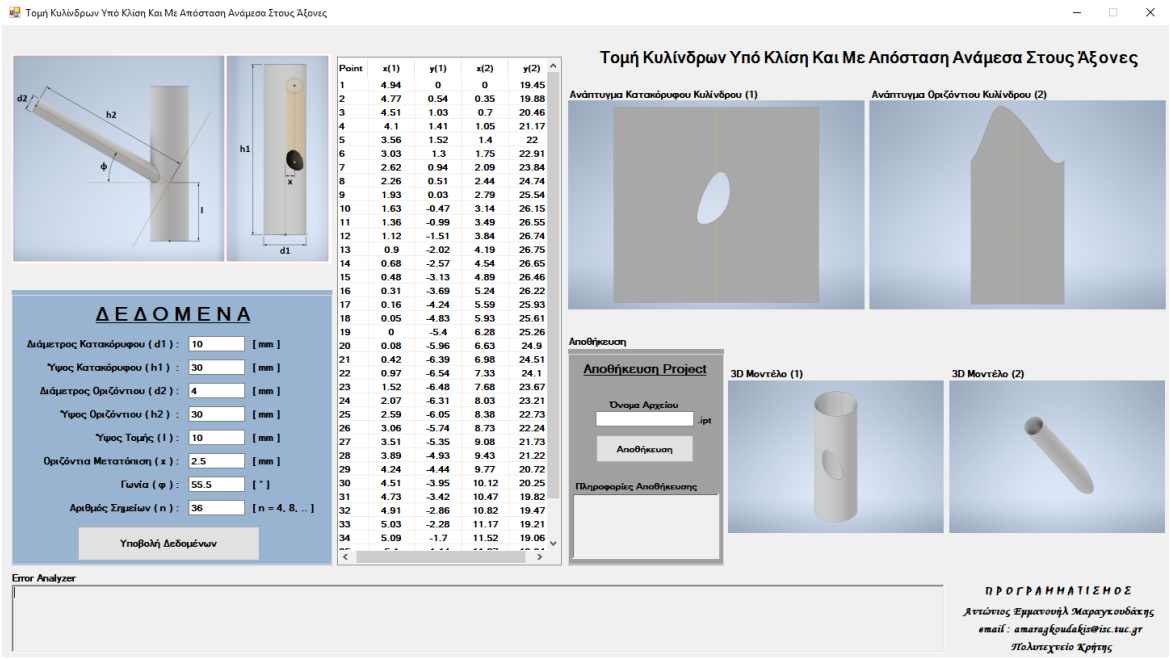
Εφόσον η εκτέλεση του API θα έχει ολοκληρωθεί, θα έχουν δημιουργηθεί κάποια παραγόμενα αρχεία τα οποία θα είναι αποθηκευμένα σε μία κρυφή τοποθεσία (path) της εφαρμογής. Όπως περιγράφηκε, σχετικές πληροφορίες έχουν παρουσιαστεί στον χρήστη εντός του GUI, αλλά η πρόσβαση του στα παραγόμενα αρχεία (.ipt, .txt, .dwg, .pdf) μπορεί να επιτευχθεί μόνο έπειτα από την αποθήκευσή τους σε μία φανερή τοποθεσία (path). Συνεπώς ένα ακόμα GroupBox (με γκρι χρώμα) είναι τοποθετημένο χαμηλά στην μέση της φόρμας με σκοπό να ομαδοποιεί τα γραφικά στοιχεία που είναι υπεύθυνα για την αποθήκευση των παραγομένων που θα έχουν δημιουργηθεί.

Αρχικά ένα TextBox περιέχεται στο εν λόγω GroupBox με σκοπό να μπορεί ο χρήστης να πληκτρολογεί εντός του το Όνομα Project που επιθυμεί.

Έπειτα ένα Button περιέχεται με σκοπό την πυροδότηση των κατάλληλων ενεργειών – εντολών, προκειμένου να αντιγραφούν τα παραγόμενα από την κρυφή τοποθεσία και να αποθηκευτούν εκ νέου σε μία φανερή. Η αποθήκευση γίνεται πάντα σε φάκελο που δημιουργείται εντός προκαθορισμένης τοποθεσίας. Προκαθορισμένη τοποθεσία αποφασίστηκε να είναι η "C:\Inventor Saves" και επόμενος η κάθε αποθήκευση δημιουργεί έναν νέο φάκελο Project εντός της τοποθεσίας αυτής στον οποίο τελικά αντιγράφονται - αποθηκεύονται τα παραγόμενα. Έτσι τελικά τα παραγόμενα μετά από μία αποθήκευση θα βρίσκονται στην τοποθεσία: "C:\Inventor Saves\Επιθυμητό Όνομα Project".

Τέλος ένα RichTextBox εμπεριέχεται με σκοπό την ενημέρωση του χρήστη (με κατάλληλα διαγνωστικά μηνύματα), σε περίπτωση που το όνομα Project υπάρχει ήδη ή σε περίπτωση που έχουν υποβληθεί μη επιτρεπτοί χαρακτήρες σε αυτό. Σε τέτοια περίπτωση λοιπόν η αποθήκευση δεν θα έχει πραγματοποιηθεί. Έτσι το RichTextBox ενημερώνει τον χρήστη για την επιτυχή ή ανεπιτυχή αποθήκευση.

Στην εικόνα που ακολουθεί παρουσιάζεται η φόρμα που περιγραφικέ (Form5), ενώ αυτή περιέχει πληροφορίες έπειτα από μία εκτέλεση του API για κάποια δεδομένα που έχουν υποβληθεί.



ΣΧΗΜΑ 3.3 – ΦΟΡΜΑ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ 5ΗΣ ΠΕΡΙΠΤΩΣΗΣ ΚΑΤΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΗΣ

Σημαντικό είναι να αναφερθεί το γεγονός ότι μία τέτοια φόρμα, όπως αυτή που αποφαίνεται ανωτέρω, υλοποιήθηκε να επανέρχεται στην άδεια αρχική της κατάσταση έπειτα από κάθε νέα υποβολή δεδομένων (ΣΧΗΜΑ 3.2). Σε περίπτωση λοιπόν όπου προκύψει μία νέα Υποβολή δεδομένων, τα προϋπάρχοντα δεδομένα του προγενέστερου υπολογισμού θα διαγράφονται. Το γεγονός αυτό μπορεί να θεωρηθεί ως reset. Όλα τα γραφικά της στοιχεία συνεπώς θα αδειάζουν έτσι ώστε οι πληροφορίες των επικείμενων υπολογισμών να μπορούν να παρουσιαστούν εντός αυτών, μόλις αυτές είναι διαθέσιμες.

Σημαντικό είναι επίσης το ότι όσο το Inventor API εκτελείται, δεν λαμβάνεται υπόψιν κανένα γεγονός (click ή enter) στην φόρμα, μέχρι την ολοκλήρωση της εκτέλεσης του. Συνεπώς μία τέτοια φόρμα μπορεί να λαμβάνει μόνο τις ακόλουθες τρεις καταστάσεις: Άδεια | Αδρανής κατά την εκτέλεση του API | Γεμάτη πληροφορίες έπειτα

Για κάθε μία εκ των πέντε περιπτώσεων Κυλινδρικών Τομών έχει αναπτυχθεί και από μία ελαφρός τροποποιημένη τέτοια φόρμα. Η φόρμα αυτή (Form5), αποτελεί την πολυπλοκότερη των υπόλοιπων τεσσάρων και επομένως επεξηγεί πλήρως όλες τις λειτουργείες που υλοποιήθηκαν. Για τον λόγο αυτό οι υπόλοιπες τέσσερεις φόρμες δεν είναι αναγκαίο να περιγραφούν.

Εφόσον μία τέτοια φόρμα είναι πλέον λειτουργική η σύνδεσή της με το Inventor API είναι το επόμενο απαραίτητο. Η σύνδεση λοιπών μίας τέτοιας φόρμας με το Inventor API περιγράφεται στο επόμενο υποκεφάλαιο.

3.2.2 Σύνδεση του GUI με το Inventor API

Ουσιαστικά το Inventor API εμπεριέχεται εντός του GUI και ενεργοποιείται έπειτα από μία Υποβολή δεδομένων. Στην συνέχεια λοιπόν του παρόντος υποκεφαλαίου θα περιγραφεί ο τρόπος που το Inventor API ενσωματώθηκε στο GUI, καθώς και θα γίνει χρήση ψευδοκώδικα που θα παρουσιάζει την εν λόγω ενσωμάτωση.

Στο σημείο αυτό χρήσιμο είναι να περιγραφούν δύο βασικοί μέθοδοι των Windows Forms. Η πρώτη εκτελεί κώδικα όσο μία φόρμα φορτώνει (loading), δηλαδή πριν αυτή παρουσιαστεί στον χρήστη. Συνεπώς εντός της μεθόδου αυτής εκτελούνται όλες οι εντολές που είναι απαραίτητο να έχουν εκτελεστεί πριν ο χρήστης αλληλεπιδράσει με μία φόρμα. Σε επίπεδο κώδικα η εν λόγω μέθοδος συντάσσεται σύμφωνα με το παρακάτω παράδειγμα.

```
Private Sub Form5_Load(sender As Object, e As EventArgs)

    Εντολές απαραίτητες πριν από την αλληλεπίδραση του χρήστη με μία φόρμα...

End Sub
```

Δεύτερη μέθοδος και πιο σημαντική είναι αυτή που ενεργοποιεί την εκτέλεση ενός συνόλου εντολών, μετά την ανίχνευση ενός γεγονότος (click ή enter). Συμφώνα με την περιγραφή που έχει γίνει, ένα Button εμπεριέχεται στην φόρμα που περιγραφικέ με σκοπό την Υποβολή των δεδομένων.

Επομένως υλοποιήθηκε όταν ανιχνευτεί click ή enter στο εν λόγω Button της φόρμας, να ενεργοποιείτε η εκτέλεση των αντίστοιχων εντολών του Inventor API, το οποίο και τελικά θα εκτελεστεί μόνο αν τα δεδομένα εισόδου είναι έγκυρα. Σύμφωνα λοιπόν με των παρακάτω ψευδοκώδικα το Inventor API υλοποιήθηκε να εκτελείται εντός της φόρμας (Form5) και εντός της μεθόδου Button_Click().

```
Public Class Form5
  Private Sub Button_Click(sender As Object, e As EventArgs) Handles Button.Click

        Reset των πληροφοριών που παρουσιάζει η φόρμα…

        Άντληση δεδομένων από τα TextBox…

        Έλεγχος εγκυρότητας δεδομένων και διάγνωση σφαλμάτων…

        If έγκυρα δεδομένα then

          ΕΝΤΟΛΕΣ INVENTOR API ΤΗΣ 5Ης ΠΕΡΙΠΤΩΣΗΣ.. (αφού εκτελούνται εντός Form5)

          Εμφάνιση υπολογισμών και εικόνων στην φόρμα και προσωρινή αποθήκευση αρχείων

        End If

    End Sub
End Class
```

Κατανοώντας λοιπόν τον ανωτέρω ψευδοκώδικα μπορεί να γίνει αντιληπτή η σύνδεση του GUI με το Inventor API τόσο σε επίπεδο σειράς εκτέλεσης όσο και σε επίπεδο επικοινωνίας. Το Inventor API εκτελείτε ουσιαστικά εντός μίας φόρμας. Έτσι η φόρμα μοιράζεται όλες τις πληροφορίες τις (δεδομένα εισόδου) με το Inventor API ενώ αντίστοιχα όλες οι πληροφορίες που το API υπολογίζει, μπορούν παράλληλα να εμφανίζονται στην αντίστοιχη φόρμα.

Φάκελος Resources

Στο σημείο αυτό, χρίζει αναφοράς το γεγονός ότι κάθε εφαρμογή που υλοποιείται εντός του περιβάλλοντος Visual Studio περιέχει έναν φάκελο με όνομα Resources, στον οποίο αποθηκεύονται από τον προγραμματιστή όλα τα απαραίτητα προκαθορισμένα αρχεία. Επομένως εκεί αποθηκεύτηκαν όλες οι προκαθορισμένες εικόνες που παρουσιάζονται στο GUI αλλά και τοποθετούνται στο Μηχανολογικό Σχέδιο από το Inventor API.

Ο Φάκελος Resources βρίσκεται εντός μιας τοποθεσίας που δεν είναι προκαθορισμένη για κάθε υπολογιστή που θα εκτελεί την εφαρμογή που υλοποιήθηκε. Έτσι η αναφορά της εν λόγω τοποθεσίας σε επίπεδο κώδικα μπορεί να γίνεται με την χρήση της εντολής που παρουσιάζεται παρακάτω. Η εντολή αυτή αντλεί την μη προκαθορισμένη πληροφορία της ακολουθίας των χαρακτήρων (String) αυτής της τοποθεσίας (path), σε οποιονδήποτε υπολογιστή και αν εκτελείται η εφαρμογή που υλοποιήθηκε.

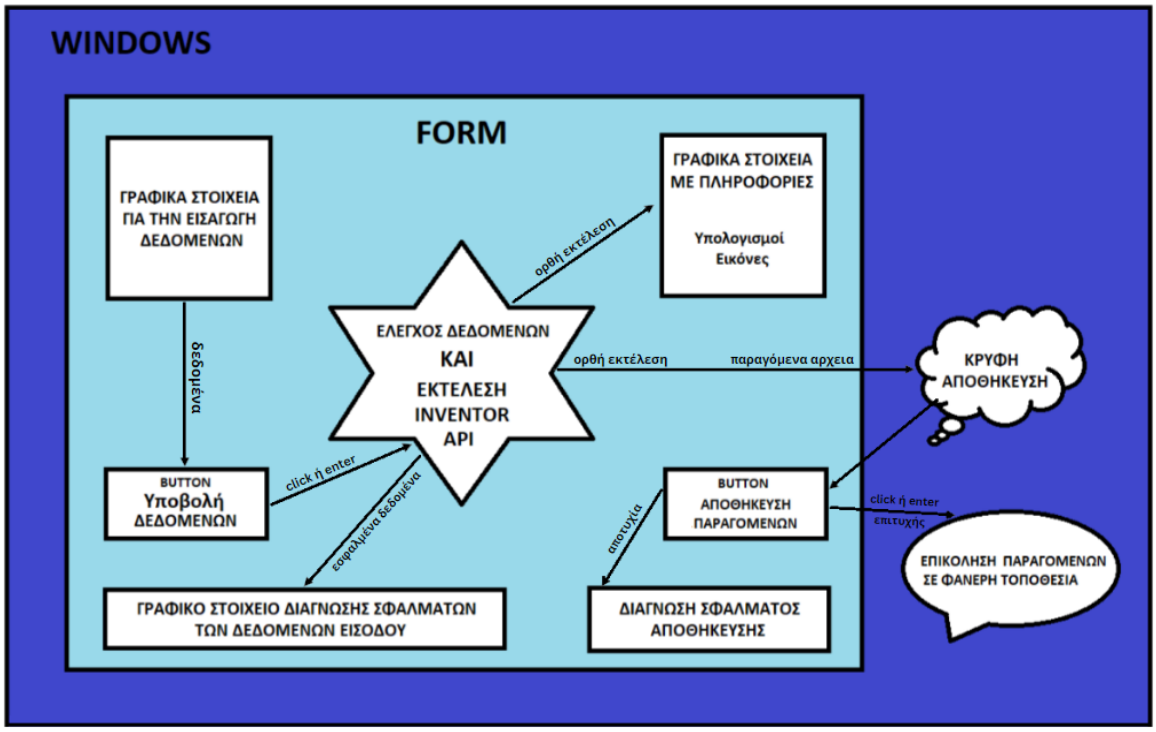
```
Dim ResourcesPath As String
ResourcesPath = My.Application.Info.DirectoryPath
```

Η τοποθεσία αυτή (Resources) είναι άγνωστη από τον χρήστη και επομένως χρησιμοποιείται για την «κρυφή» προσωρινή αποθήκευση οποιουδήποτε αρχείου πιθανώς χρειαστεί η εφαρμογή να διαχειριστεί ξανά σε επόμενη χρονική στιγμή. Επομένως είναι απαραίτητο σε επίπεδο κώδικα να διαχειριζόμαστε αρχεία που θα περιέχονται σε αυτήν την τοποθεσία για δύο λόγος. Πρώτον γιατί σε αυτήν περιέχονται οι προκαθορισμένες εικόνες που απαιτούνται τόσο από το GUI όσο και από το API, και δεύτερον γιατί στην τοποθεσία αυτή αποθηκεύονται προσωρινά (κρυφά από τον χρήστη) τα παραγόμενα του Inventor API. Σε περίπτωση νέας Υποβολής δεδομένων τα προσωρινά περιεχόμενα το εν λόγω φάκελου διαγράφονται προκειμένου να μην δημιουργείται υπερφόρτωση στην Μνήμη. Γενικότερα η εφαρμογή υλοποιήθηκε να διαχειρίζεται την τοποθεσία που περιγράφηκε όποτε είναι χρήσιμο.

Να γίνει σαφές το ότι όταν το API εκτελείται, πάντα αποθηκεύει προσωρινά τα παραγόμενα αρχεία στην εν λόγω κρυφή τοποθεσία της εφαρμογής, και αν τελικά ο χρήστης επιθυμεί την αποθήκευση αυτών, τότε σε επίπεδο λειτουργικού των Windows, τα παραγόμενα αντιγράφονται από την κρυφή τοποθεσία σε μία φανερή. Πέραν λοιπόν των πληροφοριών που παρουσιάζονται σε μία φόρμα, τα παραγόμενα αρχεία του Inventor ουσιαστικά αποθηκεύονται έμμεσα από την φόρμα αφού αυτή πρακτικά πυροδοτεί στα Windows ενέργειες Αντιγραφής και Επικόλλησης (Copy-Paste).

Έχει αναφερθεί ότι για κάθε περίπτωση Κυλινδρικών Τομών, έχει υλοποιηθεί και από μία αντίστοιχη φόρμα. Η κάθε μια φόρμα ενεργοποιεί και από ένα αντίστοιχο (τροποποιημένο) κώδικα Inventor API. Ο κώδικας που παρουσιάστηκε παραπάνω περιγράφει την σύνδεση της **5ης** φόρμας με τον αντίστοιχο κώδικα που σε επίπεδο Inventor API αυτή περιέχει. Με όμοιο τρόπο ενεργοποιείται η εκτέλεση του εκάστοτε Inventor API σε κάθε μία από τις πέντε (παρόμοιες) φόρμες και επομένως δεν απαιτείται περεταίρω επεξήγηση για αυτές.

Σύμφωνα λοιπόν με τις περιγραφές που έχουν γίνει, στην εικόνα που ακολουθεί παρουσιάζεται σχεδιαγραμματικά η βασική λειτουργεία του Inventor VBA που υλοποιήθηκε, ενώ στο επόμενο υπο-κεφάλαιο ακολουθεί η περιγραφή των ενεργειών που το Inventor API πραγματοποιεί.



ΣΧΗΜΑ 3.4 – ΔΙΑΓΡΑΜΜΑ ΛΕΙΤΟΥΡΓΕΙΑΣ ΤΟΥ INVENTOR VBA)

3.3 Περιγραφή των Ενεργειών του Inventor API

Όπως αναφέρθηκε η κάθε φόρμα υπολογισμού, ενεργοποιεί και από ένα διαφορετικό κώδικα Inventor API ο οποίος διέπεται από αντικειμενοστραφή προγραμματισμό και εκτελείτε με την βοήθεια της βιβλιοθήκης autodesk.inventor.interop.dll. Για κάθε περίπτωση Κυλινδρικών Τομών, ο αντίστοιχος κώδικας μπορεί να κατασκευάζει διαφορετικά αντικείμενα εντός του Inventor, ή να περιέχει διαφορικές συνθήκες και εντολές πράξεων. Ωστόσο οι εντολές του Inventor API εξυπηρετούν τους ίδιους σκοπούς σε κάθε περίπτωση. Συνεπώς και οι πέντε κώδικες Inventor API που υλοποιήθηκαν παρουσιάζουν κοινά βασικά βήματα με μικρές τροποποιήσεις.

Κάθε τέτοιο βασικό βήμα σε επίπεδο προγραμματισμού έχει υλοποιηθεί ως μία μέθοδο και για κάθε περίπτωση Κυλινδρικών Τομών το περιεχόμενο της αντίστοιχης μεθόδου μπορεί να είναι ελαφρός τροποποιημένο. Οι περισσότερες μέθοδοι σε επίπεδο υλοποίησης, περιέχουν μεγάλο όγκο μακροεντολών και λειτουργειών.

Στην συνέχεια λοιπόν θα περιγραφούν οι ενέργειες της κάθε μεθόδου όπου κατά γενίκευση πραγματοποιούνται σε κάθε περίπτωση Κυλινδρικών Τομών. Η εν λόγω περιγραφή θα γίνει περιεκτικά με χρήση γενικευμένων παραδειγμάτων ψευδοκώδικα. Η κάθε μέθοδος θα περιγραφεί σε βασικό επίπεδο και πρέπει να είναι σαφές ότι το πραγματικό μέγεθος των περισσότερων εξ αυτών συνήθως είναι αρκετές φορές μεγαλύτερο. Τέλος να αναφερθεί ότι οι μέθοδοι που θα περιγραφούν, καλούνται με αυστηρή σειρά κατά την εκτέλεση του API και επομένως ακολουθεί η σειρά περιγραφής τους αντίστοιχα με την σειρά όπου αυτές καλούνται - εκτελούνται.

3.3.1 Μέθοδοι στο Part Περιβάλλον του Inventor

Στο παρόν υποκεφάλαιο θα περιγραφούν όλοι οι βασικοί μέθοδοι που υλοποιήθηκαν να καλούνται και να επενεργούν εντός του περιβάλλοντος Inventor Part. Οι μέθοδοι αυτοί κυρίως αποσκοπούν στην κατασκευή των τρισδιάστατων μοντέλων, στην ανάπτυξη αυτών, στον ορθό προσανατολισμό τους καθώς και στον υπολογισμό τους. Πέραν αυτών, καθ' όλη την προαναφερθείσα διαδικασία, επιπλέον μέθοδοι συλλέγουν επιθυμητές πληροφορίες και εικόνες με σκοπό να τις προβάλλουν εντός των αντίστοιχων γραφικών στοιχείων της εκάστοτε φόρμας.

Κατασκευή 3D Μοντέλων

Στο 2ο Κεφάλαιο παρουσιάστηκε το διάγραμμα ιεραρχίας αντικειμένων που διέπει την λειτουργεία ενός Inventor API. Σύμφωνα με την περιγραφή που έγινε, αρχή των πάντων στην ιεραρχία αντικειμένων είναι το υπερ-αντικείμενο εφαρμογή (Application). Επεξηγήθηκε ο κώδικάς, ο οποίος και καταλήγει στην δήλωση ενός αντικειμένου ("partComp"), τύπου SheetMetalComponentDefinition το οποίο μπορεί να επιτρέπει την εκτέλεση ενεργειών εντός του περιβάλλοντος Sheet Metal Part όπου και πρόκειται να δημιουργηθούν τα τρισδιάστατα μοντέλα, οι κανόνες της δισδιάστατης ανάπτυξης καθώς και η ανάπτυξη των μοντέλων αυτών. Ο εν λόγω κώδικας είναι υψίστης σημασίας και για αυτό παρατίθεται ξανά παρακάτω.

```
Dim myApplication As Application
myApplication = GetObject(, "Inventor.Application")
Dim partDoc As PartDocument
Dim partComp As SheetMetalComponentDefinition
partDoc = myApplication.ActiveDocument
partDoc.SubType = "{9C464203-9BAE-11D3-8BAD-0060B0CE6BB4}"
partComp = partDoc.ComponentDefinition
Dim mySketch As PlanarSketch
mySketch = partComp.Sketches.Add(partComp.WorkPlanes.Item(2))
mySketch.SketchCircles.AddByCenterRadius(myApplication.oTG.CreatePoint2d(0,0), Diam)
```

Έχοντας λοιπόν πραγματοποιήσει την παραπάνω σύνταξη εντολών πρώτο βήμα για την κατασκευή ενός τρισδιάστατου μοντέλου είναι η κλήση μίας μεθόδου με όνομα Model_3D_Construction() που υλοποιήθηκε. Η μέθοδος αυτή παίρνει όλα τα απαραίτητα ορίσματα και δημιουργεί το εκάστοτε τρισδιάστατο μοντέλο σύμφωνα με τα δεδομένα που ο χρήστης έχει υποβάλει. Προφανώς πριν από την κλήση αυτής της μεθόδου τα δεδομένα θα έχουν αντληθεί από τα γραφικά στοιχεία TextBox που η αντίστοιχη φόρμα περιέχει. Χρησιμοποιώντας λοιπόν τα Windows Forms αυτό έγινε σύμφωνα με το παρακάτω παράδειγμα, στο οποίο για κάθε δεδομένο (TextBox) αντλείται η πληροφορία του σε μία διαφορετική μεταβλητή.

```
Height = Double.Parse(TextBox1.Text)
Diameter = Double.Parse(TextBox2.Text)
```

Η μέθοδος Model_3D_Construction() λοιπόν, υλοποιήθηκε να δημιουργεί ένα τρισδιάστατο Μοντέλο **εντός του Sheet Metal Part** περιβάλλοντος του Inventor. Η μέθοδος αυτή περιγράφεται από τον ψευδοκώδικα που ακολουθεί. Στον εν λόγω ψευδοκώδικα μπορεί να γίνει αντιληπτό το ότι όλες οι ενεργείς επενεργούν στο αντικείμενο **partComp** το οποίο και έχει δηλωθεί να είναι ένα αντικείμενο που επιτρέπει ενέργειες εντός του περιβάλλοντος Sheet Metal Part. Στο παράδειγμα λοιπόν που ακολουθεί, επιλέχτηκε λόγου χάριν και παρουσιάζεται η μέθοδος που επενεργεί με σκοπό την κατασκευή **μόνο του οριζόντιου Κυλινδρικού Τμήματος** της 4ης εκ των περιπτώσεων Κυλινδρικών Τομών (Τομή Υπό Γωνία Κυλίνδρων).

```
Sub Model_3D_Construction(partComp, Δεδομένα που έχουν Υποβληθεί από τα TextBoxes)
```

Εντολές και για την δημιουργία του κατακόρυφου κυλίνδρου

```
Δημιουργία Sketch (εντός ενός εκ των τριών προκαθορισμένων επιπέδων YZ, XY, XZ)
Dim sk1 As PlanarSketch
sk1 = partComp.Sketches.Add(partComp.WorkPlanes.Item(2))

Μέθοδος κατασκευής κύκλου εντός του sk1 PlanarSketch
sk1.SketchCircles.AddByCenterRadius(κέντρο, διάμετρος)
Επιλογή κλειστής επιφάνειας (oProfile1) μέσα από το Sketch που δημιουργήθηκε
Dim oProfile1 As Profile
oProfile1 = sk1.Profiles.AddForSurface

Προσθήκη ύψους στο προφίλ (oProfile1) του Sketch (που πριν επιλέχτηκε)
Dim oExtrude1 As ExtrudeDefinition
oExtrude1 = partComp...CreateExtrudeDefinition(oProfile1, kSurfaceOperation)
oExtrude1.SetDistanceExtent(ΥπερβένωνΤουΥψοςΤομής, kPositiveExtentDirection)
oExtrude1.SetDistanceExtentTwo(ΤιμήΥψουςΤομής)
```

Πλέον έχει δημιουργηθεί (δίχως πάχος) η επιφάνεια του κατακόρυφου κυλίνδρου

Παρομοίως συντάσσονται εντολές και για την δημιουργία του υπο γωνία κυλίνδρου

```
Δημιουργία νέου επίπεδου, υπό δεδομένη γωνία
Dim oPlane As WorkPlane
oPlane = partComp.WorkPlanes.AddByLinePlaneAndAngle(LineObject, Γωνία(rad))
Δημιουργία sketch (sk2) εντός του νέου υπο γωνία επιπέδου (oPlane)
Τοποθέτηση ενός Κύκλου στο sk2 με Διάμετρο2 (Διάμετρο Υπο Γωνία κυλίνδρου)
Επιλογή κλειστής επιφάνειας (oProfile2) που περιέχετε στο sk2
Προσθήκη ύψους (oExtrude2) στην κλειστή επιφάνεια oProfile2
```

Πλέον έχει δημιουργηθεί (δίχως πάχος) και η επιφάνεια του υπό γωνία κυλίνδρου

Έχουν λοιπόν δημιουργηθεί και οι δύο επιφανειακοί κύλινδροι σαν ομοιόμορφη επιφάνια και ακολουθεί η Μέθοδος διαχωρισμού των 3D κυλινδρικών επιφανίων σε Κυλινδρικά Τμήματα (Τομή)

```
Dim oSplit1 As SplitFeature
oSplit1 = partComp.Features.SplitFeatures.SplitFaces(WorkSurfaces, WorkPlanes)
Επιλογή της επιφάνειας του κατακόρυφου κυλίνδρου (έπειτα από την τομή)
Dim oFaceColl1 As FaceCollection
oFaceColl1 = myApplication.TransientObjects.CreateFaceCollection
oFaceColl1.Add(partComp.WorkSurfaces.Item(1)._SurfaceBody.Faces.Item(1))
```

Στην συνέχεια δίνετε μία στοιχειώδη διάσταση πάχους (0.05 mm) μόνο στον κατακόρυφο κύλινδρο ο οποίος προηγούμενος προστέθηκε στην συλλογή (oFaceColl1)

```
Dim oThicken1 As ThickenFeature
oThicken1 = partComp.Features.ThickenFeatures.Add(oFaceColl1, "0.05 mm", ...)
```

Πλέον έχει δημιουργηθεί ο κατακόρυφος κύλινδρος με μικρό (αμελητέο) πάχος και έπειτα ακολουθεί η δημιουργία κατακόρυφης σχισμής στον κατακόρυφο κύλινδρο προκειμένου να μπορεί να αναπτυχθεί το μοντέλο του (κανόνας ξεδίπλωσης)...

```
End Sub
```

Στο σημείο αυτό χρήσιμο είναι να αναφερθεί ότι κατά την χρήση της μεθόδου SetDistanceExtent() δίνεται μία διάσταση Ύψους της οποίας η κατεύθυνση θεωρείται θετική (προς τα πάνω), ενώ έπειτα στην μέθοδο SetDistanceExtentTwo() η κατεύθυνση θα θεωρηθεί ως αρνητική. Για πρακτικούς λόγους, κατά την κατασκευή των μοντέλων το Ύψος Τομής αποφασίστηκε να υλοποιείται ως σημείο μηδέν, προκειμένου στο σημείο μηδέν να συμβεί και η δημιουργία του υπο γωνία επιπέδου και κυλίνδρου, και κατ' επέκτασιν η και η τομή των δύο κυλινδρικών τμημάτων. Για τον λόγο αυτό λοιπόν το ύψος τομής δόθηκε να περιέχετε με αρνητική κατεύθυνση κάτω από το σημείο της τομής (μηδέν): SetDistanceExtentTwo(ΤιμήΥψουςΤομής).

Σημαντικό είναι επίσης το γεγονός το ότι η μέθοδος υλοποιήθηκε αρχικά να κατασκευάζει Κυλινδρικές Επιφάνειες τις οποίες κατόπιν διαχωρίζει χρησιμοποιώντας την μέθοδο SplitFaces(). Τελικά προσδίδεται πάχος (ύπαρξη) μόνο στο κατακόρυφο Κυλινδρικό Τμήμα. Πρέπει να είναι λοιπόν σαφές το ότι μετά την εκτέλεσή της, η μέθοδος θα έχει δημιουργήσει μόνο το τρισδιάστατο μοντέλο του κατακόρυφου κυλίνδρου, αφού μόνο σε αυτό προσδίδεται πάχος (ύπαρξη). Όλα τα υπόλοιπα αντικείμενα είναι είτε σκίτσα είτε επιφάνειες και δεν καταλαμβάνουν χώρο.

Επομένως μία παρόμοια μέθοδος Model_3D_Construction**_2**() έχει υλοποιηθεί για να δημιουργεί το τρισδιάστατο μοντέλο του (2ου) υπό γωνία Κυλινδρικού Τμήματος. Η Model_3D_Construction_2() αντίστοιχα προσδίδει πάχος (ύπαρξή) μόνο στο υπο γωνία Κυλινδρικό Τμήμα και πουθενά αλλού. Επίσης σημαντικό είναι να αναφερθεί το ότι η μέθοδος αυτή δημιουργεί το ($2^{ο}$) τρισδιάστατο Κυλινδρικό Τμήμα εντός ενός νέου αντικειμένου **partComp2** το οποίο ουσιαστικά υλοποιήθηκε να αποτελεί ένα νέο αρχείο περιβάλλοντος Sheet Metal Part, το οποίο είναι ανεξάρτητο του προηγούμενου. Σύμφωνα με τον ψευδοκώδικα που ακολουθεί παρουσιάζονται οι διαφορές που φέρει η εν λόγω μέθοδος σε σχέση με την προηγούμενη.

```
Sub Model_3D_Construction_2(partComp2, Δεδομένα που έχουν Υποβληθεί)

        ...Ομοίως με πριν μέχρι και των διαχωρισμό σε δύο κυλινδρικά τμήματα (Split)

        Επιλογή της επιφάνειας του υπό Γωνία Κυλινδρικού Τμήματος (.Item(2))
        Dim oFaceColl2 As FaceCollection
        oFaceColl2 = myApplication.TransientObjects.CreateFaceCollection
        oFaceColl2.Add(partComp2.WorkSurfaces.Item(2)._SurfaceBody.Faces.Item(1))

        Πλέον δίνουμε μία στοιχειώδη διάσταση πάχους στον κατακόρυφο κύλινδρο
        Dim oThicken1 As ThickenFeature
        oThicken1 = partComp2.Features.ThickenFeatures.Add(oFaceColl2, "0.05 mm",...)

        Πλέον έχει δημιουργηθεί ο υπό γωνία κύλινδρος με μικρό (0.05 mm) πάχος και
        έπειτα ακολουθεί η δημιουργία σχισμής στον κύλινδρο αυτό, προκειμένου το μοντέλο
        του να είναι σε θέση να αναπτυχθεί αργότερα (κανόνας ξεδίπλωσης)...

End Sub
```

Ουσιαστικά όλες οι περιπτώσεις Κυλινδρικών Τομών που υλοποιήθηκαν περιέχουν δύο Κυλινδρικά Τμήματα. Εξαίρεση αποτελεί η πρώτη περίπτωση όπου κατασκευάζει και υπολογίζεται το τρισδιάστατο μοντέλο της τομής ενός Κυλίνδρου με ένα Υπο Γωνία Επίπεδο. Στις υπόλοιπες τέσσερις περιπτώσεις τέμνονται δύο Κυλινδρικά Τμήματα και επομένως για τον λόγο αυτό αναφέρουμε την ορολογία σε πληθυντικό αριθμό, και δεν λέμε Κυλινδρική Τομή. Κάθε περίπτωση περιέχει δύο Κυλινδρικά Τμήματα και για τον λόγο αυτό, δύο είναι και οι τομές που προκύπτουν όταν σκεφτόμαστε διαχωρισμένο το ένα Κυλινδρικό Τμήμα από το άλλο.

Όπως μπορεί να φανεί στον ψευδοκώδικα που παρουσιάστηκε, η μέθοδος Model_3D_Construction_2() δημιουργεί το τρισδιάστατο μοντέλο του υπό γωνία Κυλινδρικού Τμήματος, εντός ενός νέου αντικειμένου partComp2 το οποίο ουσιαστικά αποτελεί ένα νέο αρχείο του περιβάλλοντος Sheet Metal Part, το οποίο είναι ανεξάρτητο του προηγούμενου. Γενικότερα λοιπόν κάθε Κυλινδρικό Τμήμα δημιουργείται πάντα σε ένα νέο - και ανεξάρτητο από τα προηγούμενα - αρχείο τύπου Sheet Metal Part.

Δεδομένου ότι περιέχονται δύο Κυλινδρικά Τμήματα σε κάθε μία εκ των πέντε περιπτώσεων πλην της πρώτης, συνολικά εννέα παρεμφερείς τύποι Κυλινδρικών Τμημάτων κατασκευάζονται και υπολογίζονται από την εφαρμογή που υλοποιήθηκε. Στις περιπτώσεις που δύο Κυλινδρικά Τμήματα τέμνονται, αυτά σε επίπεδο κώδικα κατασκευάζονται, υπολογίζονται, και παρουσιάζονται σε διαφορετικά και ανεξάρτητα αρχεία. Επομένως η κάθε βασική μέθοδος εξ αυτών που παρουσιάζονται έχει υλοποιηθεί (ελαφρώς τροποποιημένα) έως και εννέα φορές. Μία για κάθε περίπτωση Κυλινδρικού Τμήματος.

Για κάθε επόμενη μέθοδο που θα περιγραφεί στην συνέχεια θα θεωρείται ότι αυτή επενεργεί με σκοπό να διαχειρίζεται (κατασκευάζει, υπολογίζει, παρουσιάζει) μόνο ένα Κυλινδρικό Τμήμα. Επομένως θα γίνεται αναφορά της γενικότερης λειτουργείας κάθε μεθόδου για ένα αόριστο Κυλινδρικό Τμήμα. Αυτό αφού η εξειδικευμένη ανάλυση όλων των μεθόδων για την κάθε περίπτωση απαιτεί τεράστιο όγκο περιγραφών και παραδειγμάτων.

Κλείνοντας λοιπόν την περιγραφή της μεθόδου Model_3D_Construction(), γνωρίζουμε ότι αυτή έπειτα από την εκτέλεσή της θα έχει δημιουργήσει το τρισδιάστατο μοντέλο ενός Κυλινδρικού Τμήματος, όπως και θα έχει υλοποιήσει μια σχισμή σε αυτό ως απαραίτητο κανόνα ξεδίπλωσης.

Μέθοδος Screenshot_3D()

Στην συνέχεια σειρά έχει η εκτέλεση της μεθόδου ScreenShot_3D() της οποίας η λειτουργεία και υλοποιήθηκε. Όπως έχει αναφερθεί όσο το API εκτελείται, σχετικές πληροφορίες παρουσιάζονται στο GUI όπου ο χρήστης αλληλεπιδρά. Στο συγκεκριμένο παράδειγμα λοιπόν, όταν το τρισδιάστατο μοντέλο ενός Κυλινδρικού Τμήματος έχει ολοκληρωθεί, το API τραβάει screenshots το μοντέλο αυτό και έπειτα το προβάλει στον χρήστη εντός του αντίστοιχου PictureBox που περιέχεται στην ενεργή φόρμα που αυτός έχει υποβάλλει τα δεδομένα προς υπολογισμό. Η μέθοδος ScreenShot_3D() λοιπόν, είναι υπεύθυνη αυτής της λειτουργείας και ενεργεί σε επίπεδο εντολών σύμφωνα με το παρακάτω παράδειγμα, στο οποίο το API χειρίζεται την κάμερα του Inventor. Έχοντας λοιπόν ήδη παράξει ένα τρισδιάστατο μοντέλο ενός Κυλινδρικού Τμήματος, το API επενεργεί στην κάμερα του Inventor και την κεντράρει στο αντικείμενο αυτό, ενώ στην συνέχεια κρύβει και το γραφικό στοιχείο του συστήματος αξόνων (3DIndicator). Έπειτα αποθηκεύει την προβολή της κάμερας με μορφή εικόνας (.bmp) στον «κρυφό» φάκελο Resources ενώ στην συνέχεια προβάλει την αποθηκευμένη αυτή εικόνα στο εκάστοτε PictureBox που έχει δοθεί ως όρισμα στην μέθοδο. Τέλος επαναφέρει πάλι το σύστημα αξόνων σε ορατή κατάσταση, ως αυτό προϋπήρχε. Η προπεριγραφείσα λοιπόν λειτουργεία της μεθόδου Screenshot_3D(), αντιστοιχεί στο παρακάτω παράδειγμα εντολών.

```
Sub Screenshot_3D (myApplication, ResourcesPath, myFileName, myPictureBox)

	Dim myCamera As Inventor.Camera
	myCamera = myApplication.ActiveView.Camera
	myApplication.DisplayOptions.Show3DIndicator = False
	myCamera.ViewOrientationType = 10760
	myCamera.Fit()
	myCamera.ApplyWithoutTransition()

	myCamera.SaveAsBitmap(ResourcesPath & myFileName & ".bmp",διάσταση1,διάσταση2)

	myPictureBox.Image = Image.FromFile(myPath & myFileName & ".bmp")
	myApplication.DisplayOptions.Show3DIndicator = True

End Sub
```

Μέθοδος Create_Flatt_Pattern()

Έχοντας εκτελεστεί η μέθοδος Model_3D_Construction(), το τρισδιάστατο μοντέλο ενός Κυλινδρικού Τμήματος θα έχει δημιουργηθεί περιέχοντας ήδη το χαρακτηριστικό της σχισμής που του επιτρέπει να αναπτυχθεί στον δισδιάστατο χώρο. Έτσι η μέθοδος Create_Flatt_Pattern() υλοποιήθηκε να επενεργεί στο σημείο αυτό με σκοπό την ανάπτυξη του τρισδιάστατου Κυλινδρικού Τμήματος σύμφωνα με το παρακάτω παράδειγμα. Αρχικά επιλέγετε το κατάλληλο πρόσωπο (Face) του κυλίνδρου ως ASideDefinition (επιφάνεια εσωτερική ή εξωτερική). Έπειτα με χρήση της έτοιμης μεθόδου Unfold() ακολουθεί η ανάπτυξη με το εν λόγω πρόσωπο να παρουσιάζεται στην κορυφή του αναπτύγματος.

```
Public Sub Create_Flatt_Pattern(partComp)

        Dim myface As Face
        myface = partComp.SurfaceBodies(1).Faces.Item(δείκτης)
        partComp.ASideDefinitions.Add(myface)
        partComp.Unfold()

End Sub
```

Πλέον λοιπόν θα έχει δημιουργηθεί ένα αντικείμενο (Flatt Pattern) το οποίο θα περιέχει το ανάπτυγμά ξαπλωμένο στον Τρισδιάστατο εικονικό χώρο του Inventor. Το οποιοδήποτε ανάπτυγμα, ως τρίτη διάσταση (ύψος) θα περιέχει το σχεδόν αμελητέο πάχος (0.05 mm) που προκαθορισμένα το API πάντα προσδίδει στο κάθε Κυλινδρικό Τμήμα.

Μέθοδος Rotate_Flat_Pattern()

Στο σημείο αυτό το αντικείμενο του αναπτύγματος (Flatt Pattern) έχει δημιουργηθεί από την εντολή partComp.Unfold(), προκαθορισμένα με την μεγαλύτερη ακμή του να βρίσκεται ξαπλωμένη στον άξονα X. Αυτό έχει σαν αποτέλεσμα (ανάλογα με την είσοδο) πολλές φορές το ανάπτυγμα να μην τοποθετείται καθ' ύψος, αλλά ξαπλωμένο. Επομένως η μέθοδος Rotate_Flat_Pattern() έχει υλοποιηθεί στο σημείο αυτό με σκοπό να προσανατολίζει το ανάπτυγμα ώστε η βάση του να εξελίσσεται πάντα παράλληλα στον άξονα X και το ύψος στον άξονα Y. Σύμφωνα με το παράδειγμα που ακολουθεί, ανάλογα με την αρχική τοποθέτηση εκτελούνται και οι απαραίτητες μακροεντολές προσανατολισμού (Orientation). Οι μακροεντολές αυτές παρουσιάζονται στο

παράδειγμα εντός της μεθόδου και δεν εκτελούνται όλες πάντα, αλλά σύμφωνα με κατάλληλους ελέγχους, εκτελούνται όποιες είναι απαραίτητες κάθε φορά.

```
Public Sub Rotate_Flat_Pattern(myFlatPattern, Δεδομένα Ελέγχου)

	Έλεγχος Δεδομένων για την επιλογή των κατάλληλων εντολών προσανατολισμού
	Ανάλογα λοιπόν με τον έλεγχο της αρχικής τοποθέτησης, όποιες από τις
	παρακάτω εντολές είναι απαραίτητες εκτελούνται έτσι ώστε να γίνει η
	κατάλληλη περιστροφή σύμφωνα με τον εκάστοτε έλεγχο.

	myFlatPattern.FlatPatternOrientations.ActiveFlatPatternOrientation.Alignmen
	tType = AlignmentTypeEnum.kVerticalAlignment

	myFlatPattern.FlatPatternOrientations.ActiveFlatPatternOrientation.FlipAlig
	nmentAxis = True

	myFlatPattern.FlatPatternOrientations.ActiveFlatPatternOrientation.FlipBase
	Face = True

End Sub
```

Μέθοδος Screenshot_Fp()

Πλέον το ανάπτυγμα (αντικείμενο Flatt Pattern) θα έχει την σωστή τοποθέτηση με την βάση του να είναι ξαπλωμένη παράλληλα στον άξονα X και το ύψος του να εξελίσσεται προς τα πάνω παράλληλα με τον άξονα Y. Έτσι στο σημείο αυτό η μέθοδος Screenshot_Flatt_Pattern() έχει υλοποιηθεί να αποθηκεύει ένα στιγμιότυπο του αναπτύγματος και να το προβάλει στην διεπαφή του χρήστη. Η μέθοδος αυτή ενεργεί παρόμοια με την μέθοδο Screenshot_3D() με βασική διαφορά το γεγονός ότι δέχεται ως όρισμα το αντικείμενο Flatt Pattern. Σαν πρώτο βήμα λοιπόν ενεργοποιεί την επεξεργασία αυτού του αντικειμένου προκειμένου να διαφαίνεται αυτό εντός της κάμερας του Inventor, έναντι του τρισδιάστατου Κυλινδρικού Τμήματος. Η λειτουργεία αυτή σε επίπεδο ψευδοκώδικα παρατίθεται στην συνέχεια.

```
Sub Screenshot_Fp(myFlatPattern, myApplication, ResourcesPath, myFileName, myPictureBox)

	myFlatPattern.Edit()
	Dim myCamera As Inventor.Camera
	myCamera = myApplication.ActiveView.Camera
	myApplication.DisplayOptions.Show3DIndicator = False
	myCamera.ViewOrientationType = 10760
	myCamera.Fit()
	myCamera.ApplyWithoutTransition()
	myCamera.SaveAsBitmap(ResourcesPath & myFileName & ".bmp", διάσταση1, διάσταση2)
```

```
        myPictureBox.Image = Image.FromFile(ResourcesPath & myFileName & ".bmp")
        myApplication.DisplayOptions.Show3DIndicator = True

End Sub
```

Calculate()

Εφόσον ένα Κυλινδρικό Τμήμα έχει αναπτυχθεί σημαντικό πρόβλημα είναι ο υπολογισμός των σημείων που περιέχονται στις καμπύλες. Επομένως η μέθοδος Calculate() υλοποιήθηκε με σκοπό τον υπολογισμό αυτό. Η μέθοδος αυτή υλοποιήθηκε να παίρνει ως όρισμα το αντικείμενο του αναπτύγματος Flatt Pattern, τον αριθμό υπολογισμών (Num) που επιθυμεί ο χρήστης, και την περιστροφή που έχει γίνει στο ανάπτυγμα. Με κατάλληλη χρήση αυτών των ορισμάτων λοιπόν υπολογίζει και επιστρέφει δύο λίστες με σημεία, μία λίστα X() και μία Y() μεγέθους Num στοιχείων. Η Μέθοδος αυτή διαχειρίζεται τις καμπύλες (Edges) που περιέχονται στο αντικείμενο του Flatt Pattern. Αν και στα τελικά αποτελέσματα το κάθε ανάπτυγμα φαίνεται να περιέχει μία μόνο ενιαία καμπύλη, αυτό σε επίπεδο αντικειμένων εντός του Inventor δεν ισχύει πάντα.

Σε επίπεδο υλοποίησης από περίπτωση σε περίπτωση, σύμφωνα με την σειρά κατασκευής των μοντέλων και την τοποθέτηση της σχισμής (κανόνας ανάπτυξης), η φαινομενικά ενιαία καμπύλη ενός αναπτύγματος παρατηρήθηκε να διαχωρίζεται μέχρι και σε τρία διαφορετικά αντικείμενα τύπου Edge. Επομένως σε επίπεδο κώδικα η μέθοδος Calculate() αναλύει κάθε φορά διαφορετικό αριθμό τμημάτων καμπύλης (Edge), και έτσι η μέθοδος αυτή παρουσιάζει αρκετές τροποποιήσεις ανάλογα με την κάθε περίπτωση Αναπτύγματος Κυλινδρικού Τμήματος που μπορεί να υπολογίζει. Στον ψευδοκώδικα λοιπόν που ακολουθεί παρουσιάζεται η βασική λειτουργεία αυτής της μεθόδου που αναλύει μία μόνο καμπύλη και υπολογίζει όλα τα σημεία της (Num).

```
Sub Calculate(myFlattPattern, Num, Περιστροφή, X(), Y(), ΔίκτηςΚατάλληλουEdge)

    Dim fpBody As SurfaceBody = myFlattPattern.SurfaceBodies(1)
    Dim myEdges As Edges = fpBody.Edges
    Dim myEdge As Edge
    myEdge = myEdges.Item(ΔίκτηςΚατάλληλουEdge)

    Dim oCurveEval As CurveEvaluator
    oCurveEval = myEdge.Evaluator

    Dim dMinParam As Double
    Dim dMaxParam As Double
    Call oCurveEval.GetParamExtents(dMinParam, dMaxParam)
```

```
    Dim currentParam, adParam(0), adPoints(1) As Double
    Dim i As Integer
    For i = 0 To Num - 1

        CurrentParam = dMinParam + ((dMaxParam - dMinParam)/(Num - 1)) * i
        adParam(0) = currentParam
        Call oCurveEval.GetPointAtParam(adParam, adPoints)

        Εντολή If αποφασίζει τις πράξεις και την χρήση των τιμών adPoints(0) και
        adPoints(1) ανάλογα με την θέση του κάθε Edge σε σχέση με τον αρχικό
        προσανατολισμό του αναπτύγματος
        X(i) = τροποποιημένες πράξεις της τιμής adPoints(0) ή adPoints(1)
        Y(i) = τροποποιημένες πράξεις της τιμής adPoints(0) ή adPoints(1)
    Next

End Sub
```

Σύμφωνα λοιπόν με τον ψευδοκώδικα αυτό, η εν λόγω μέθοδος αρχικά επιλέγει το κατάλληλο Edge που πρέπει να αναλυθεί. Έπειτα για το δεδομένο Edge αντλείται το αντικείμενο τύπου CurveEvaluator που το διέπει. Το αντικείμενο αυτό λαμβάνει το έγκυρο εύρος παραμέτρων ενός Edge. Έπειτα έχοντας λάβει λοιπόν την μέγιστη και ελάχιστη τιμή εύρους, εντός κάθε επανάληψης υπολογίζεται μία παροντική τιμή currentParam. Η τιμή αυτή προχωράει με σταθερό βήμα (ανάμεσα στο εύρος) κάθε φορά, ξεκινώντας από την τιμή dMinParam στην πρώτη επανάληψη και καταλήγοντας με τιμή dMaxParam στην τελευταία.

Η παράμετρος της κάθε επανάληψης currentParam, για λόγους του Inventor, απαιτείται να δοθεί ως όρισμα στην μέθοδο GetPointAtParam() και υποχρεωτικά με χρήση μίας λίστας "adParam(0) ". Όπως έχει αναφερθεί ένα αντικείμενο CurveEvaluator διέπει μία καμπύλη. Επομένως με την χρήση της μεθόδου oCurveEval.GetPointAtParam(adParam, adPoints) επιστέφονται δύο τιμές στην λίστα adPoints(1) σύμφωνα με το currentParam της κάθε επανάληψης. Οι τιμές τις λίστας αυτής πλέων αναφέρονται σε τιμές X και Y του καρτεσιανού συστήματος αξόνων.

Γενικότερα οι παράμετροι που περιγράφουν μία καμπύλη εντός του Inventor διέπονται από μαθηματικά τρισδιάστατης λογικής και δεν αντιπροσωπεύουν άμεσα τιμές (x,y,z) εντός του τρισδιάστατου συστήματος αξόνων. Το εύρος [dMinParam, dMaxParam] συνεπώς δεν αντιπροσωπεύει τιμές X,Y αλλά είναι ένα εύρος που εξαρτάται από την γεωμετρική κατασκευή της κάθε καμπύλης. Επομένως ένα σταθερό βήμα ανάμεσα στο εύρος αυτό, όπως το currentParam, δεν συνεπάγεται πάντα σταθερό βήμα των τιμών στον άξονα X, αλλά μπορεί να συνεπάγεται ανάλογα με τον τύπο της κάθε καμπύλης είτε σταθερό βήμα στο μήκος της, είτε στην γωνία της. Επομένως οι τελικές τιμές X,Y που υπολογίζονται μέσω της παραμέτρου currentParam δεν διέπονται πάντα

από σταθερό βήμα ανάμεσα στους υπολογισμούς των τιμών X. Η απώλεια αυτού του σταθερού βήματος στις τιμές που υπολογίστηκαν δεν ήταν πρόβλημα στους υπολογισμούς μας, αφού και ένα σταθερό βήμα δειγματοληψίας κατά σταθερή γωνία είτε κατά σταθερό μήκος καμπύλης, είναι απολύτως αποδεκτά βήματα μίας δειγματοληψίας σημείων ενός Μηχανολογικού Σχεδίου.

Παρόλα αυτά μία τροποποιημένη μέθοδος υλοποιήθηκε με σκοπό των υπολογισμό (Num) σημείων με σταθερό βήμα στους υπολογισμούς του X άξονα. Η τροποποιημένη αυτή μέθοδος υπολογισμού (μιας και υλοποιήθηκε) εφαρμόστηκε στην $4^{η}$ και $5^{η}$ περίπτωση Κυλινδρικών Τομών.

Προκειμένου οι υπολογισμοί να γίνονται με σταθερό βήμα στον υπολογισμό των X τιμών, και δίχως οι παράμετροι του Inventor να συμβαδίζουν με αυτό, έπρεπε να υπολογίζονται πολλές τιμές μέχρις ότου αυτό να συμβεί. Σύμφωνα λοιπόν με το συνολικό μήκος της βάσης του κάθε αναπτύγματος αντιπροσωπεύεται το εύρος των X τιμών. Δηλαδή το εύρος των X είναι το διάστημα [0, π*Διάμετρος].

Επομένως με σταθερό βήμα στο διάστημα αυτό οι τιμές των επιθυμητών X αρχικά υπολογίζονται και επομένως μένει να υπολογιστούν οι αντίστοιχες Y τιμές. Αφού οι παράμετροι του CurveEvaluator βασίζονται σε άλλα σταθερά βήματα (γωνίας ή μήκους καμπύλης), έπρεπε να μη οδηγείται με σταθερό βήμα το currentParam. Επομένως στην τροποποιημένη αυτή μέθοδο, υλοποιήθηκε για κάθε υπολογισμό, με χρήση μίας επιπλέον επανάληψης να υπολογίζονται συνεχώς τιμές X,Y μέχρις ότου να υπολογιστεί μία τιμή που περιέχει ένα επιθυμητό X τουλάχιστον κατά προσέγγιση 3ου δεκαδικού ψηφίου. Όταν αυτό έχει συμβεί μπορούμε να μάθουμε το Y που έχει επιστραφεί παρέα με αυτό το αντίστοιχο X. Αυτό επαναλαμβάνεται για κάθε επόμενη επιθυμητή τιμή του X. Η μέθοδος αυτή θα ήταν πολύ αργή αφού ο υπολογισμός κάθε σημείου απαιτεί από μία επιπλέον επανάληψη που προχωράει με ένα πολύ μικρό βήμα, προκειμένου να υπολογίζει συνεχώς (χιλιάδες) σημεία μέχρις ότου να βρεθεί ένα που να αντιστοιχεί σε μία επιθυμητή τιμή X με ακρίβεια 3ου δεκαδικού ψηφίου. Ωστόσο αυτό λύθηκε και η εκτέλεση έγινε υπερβολικά πιο γρήγορη (n*log(n)), με χρήση της μεθόδου διαίρει και βασίλευε σύμφωνα με το παρακάτω παράδειγμα ψευδοκώδικα.

Ο ψευδοκώδικας της μεθόδου Calculate2() παρατίθεται παρακάτω ο οποίος και επιτρέπει να γίνει αντιληπτή η χρήση της μεθόδου διαίρει και βασίλευε. Στο εν λόγω παράδειγμα η παράμετρος currentParam αρχικοποιείται με τιμή ίση με το μισό εύρος [dMinParam, dMaxParam] και υπολογίζει τις αντίστοιχες τιμές X,Y. Εν συνέχεια για κάθε υπολογισμένη τιμή του X, όταν ο υπολογισμός του ξεπερνάει την επιθυμητή τιμή του, ανάλογα με την φορά που αυτή ξεπεράστηκε επαναπροσδιορίζεται η παράμετρος

currentParam στην οποία προστίθεται ή αφαιρείται πλέον το μισό του προηγούμενου βήματός της. Η τιμή του currentParam λοιπόν, αυξάνει ή μειώνει με το μισό του προηγούμενου βήματος και επομένως η τιμή του currentParam τελικά προσεγγίζει πολύ γρήγορα την τιμή της, που αντιστοιχεί στο επιθυμητό X, και κατ' επέκτασιν και στο αντιστοίχως άγνωστο επιθυμητό Y. Ο εν λόγω κώδικας παρατίθεται παρακάτω.

```
Sub Calculate2(myFlattPattern, Num, Περιστροφή, X(), Y(), ΔίκτηςΚατάλληλουEdge)

    Dim fpBody As SurfaceBody = myFlattPattern.SurfaceBodies(1)
    Dim myEdges As Edges = fpBody.Edges
    Dim myEdge As Edge = myEdges.Item(ΔίκτηςΚατάλληλουEdge)

    Dim dMinParam, dMaxParam As Double
    Dim oCurveEval As CurveEvaluator = myEdge.Evaluator
    Call oCurveEval.GetParamExtents(dMinParam, dMaxParam)

    Dim i As Integer
    For i = 0 To Num - 1

        Dim desirableX As Double = ((math.pi * Διάμετρος) /(Num - 1)) * i
        Dim currentParam As Double = ((dMaxParam - dMinParam) /2)
        Dim step As Double = ((dMaxParam - dMinParam) /2)
        Dim previousX As Double = 0

        Dim currentParam, adParam(0),adPoints(2) As Double
        Do
            currentParam = dMinParam + ((dMaxParam - dMinParam)/(Num - 1)) * i
            adParam(0) = currentParam
            Call oCurveEval.GetPointAtParam(adParam, adPoints)

            Εντολή If αποφασίζει για τις πράξεις και την χρήση των τιμών
            adPoints(0) και adPoints(1) ανάλογα με την θέση του κάθε Edge σε
            σχέση με τον αρχικό προσανατολισμό του αναπτύγματος
            X(i) = τροποποιημένες πράξεις με adPoints(0) ή adPoints(1)
            Y(i) = τροποποιημένες πράξεις με adPoints(0) ή adPoints(1)

            Επαναπροσδιορισμός τις επόμενης παραμέτρου currentParam
            If X(i) < desirableX And previousX < desirableX Then
                currentParam = currentParam + step
            ElseIf X(i) > desirableX And previousX > desirableX Then
                currentParam = currentParam - step
            ElseIf X(i) > desirableX And previousX < desirableX Then
                step = step/2
                currentParam = currentParam - step
            ElseIf X(i) < desirableX And previousX > desirableX Then
                step = step/2
                currentParam = currentParam + step
            End If
            previousX = X(i)

        Loop Until Format(X(i), "0.000") = Format(desirableX, "0.000")

    Next
End Sub
```

Κλείνοντας να συνοψίσουμε το ότι γενικότερα, αφού έχει ολοκληρωθεί η εκτέλεσή μίας μεθόδου Culculate(), αυτή θα έχει επιστρέψει δύο λίστες xPoints(), yPoints(). Οι λίστες αυτές θα είναι μεγέθους Num και θα περιέχουν τους επιθυμητούς υπολογισμούς της καμπύλης του αναπτύγματος ενός Κυλινδρικού Τμήματος. Στην συνέχεια οι τιμές αυτές θα πρέπει να παρουσιαστούν στην ενεργή φόρμα που ο χρήστης αλληλεπιδρά.

Fill_Lists()

Θεωρώντας ότι έχουν ολοκληρωθεί οι υπολογισμοί για μία περίπτωση με δύο Κυλινδρικά Τμήματα, σημαίνει ότι θα έχει ολοκληρωθεί η εκτέλεση δύο Μεθόδων υπολογισμού σημείων Calculate(), μία για κάθε Κυλινδρικό Τμήμα. Αυτό σημαίνει ότι θα έχουν υπολογιστεί τέσσερεις λίστες με τιμές (X1(), Y1(), X2(), Y2()). Στο σημείο αυτό η μέθοδος Fill_Lists() έχει υλοποιηθεί να παίρνει ως όρισμα τις λίστες αυτές και να τις προβάλει στο ListBox της εκάστοτε ενεργής μάσκας. Η μέθοδος αυτή έχει ως επιπλέον ορίσματα των αριθμό υπολογισμών (Num) και το επιθυμητό αντικείμενο myListView εντός του οποίου θέλουμε να προβάλουμε τις λίστες με τους υπολογισμούς. Η μέθοδος αυτή σε επίπεδο εντολών παρουσιάζεται παρακάτω.

```
Sub Fill_List(Num, X1(), Y1(), X2(), Y2(), myListView)

	Dim oListViewItem As ListViewItem

	For i = 0 To Num - 1
		oListViewItem = myListView.Items.Add(i + 1)
		oListViewItem.SubItems.Add(X1(i))
		oListViewItem.SubItems.Add(Y1(i))
		oListViewItem.SubItems.Add(X2(i))
		oListViewItem.SubItems.Add(Y2(i))
	Next

	oListViewItem = myListView.Items.Add(Num + 1)
	oListViewItem.SubItems.Add("-")
	oListViewItem.SubItems.Add("-")
	oListViewItem.SubItems.Add(X2(Num))
	oListViewItem.SubItems.Add(Y2(Num))

End Sub
```

Τα κάθετα ή υπό γωνία Κυλινδρικά Τμήματα περιέχουν πάντα μία επιπλέον τιμή υπολογισμών (Num + 1), εφόσον δεν περιέχουν κλειστή καμπύλη στα αναπτύγματά τους. Δηλαδή η καμπύλη που περιέχουν παρουσιάζει διαφορετική αρχή από τέλος και

επομένως ένα παραπάνω σημείο υπολογίζεται. Κατά συνέπεια λοιπόν οι επιπλέον τιμές (X και Y) του τελευταίου σημείου τους, περνάνε σαν τελευταίες τιμές του ListView έπειτα από το τέλος της επανάληψης. Έπειτα από την επαναληπτική διαδικασία λοιπόν, στις πρώτες δύο στήλες του ListView τοποθετείται ο χαρακτήρας της παύλας "-", εφόσον αυτές περιέχουν έναν λιγότερο υπολογισμένο σημείο.

Close_And_Save_File()

Στο σημείο αυτό, έπειτα από την εκτέλεση όλων των μεθόδων που έχουν αναφερθεί, θα έχουν ολοκληρωθεί όλες οι απαραίτητες ενέργειες εντός του περιβάλλοντος Inventor Part, καθώς και θα έχουν προβληθεί στην φόρμα που ο χρήστης αλληλεπιδρά όλες οι απαραίτητες πληροφορίες. Δηλαδή θα έχουν παρουσιαστεί οι υπολογισμένες τιμές των καμπυλών, οι δύο εικόνες των Τρισδιάστατων Μοντέλων των Κυλινδρικών Τμημάτων καθώς και οι δύο εικόνες των αναπτυγμάτων αυτών. Σημαντικό στο σημείο αυτό είναι λοιπόν να αποθηκεύονται τα αρχεία Sheet Metal Part όπου περιέχουν αυτά τα τρισδιάστατα μοντέλα και αναπτύγματα, για δύο λόγους. Πρώτον τα αρχεία αυτά (.ipt) αποτελούν μέρος των παραγομένων του API προς αποθήκευση από τον χρήστη. Δεύτερον τα ίδια αρχεία είναι πάντα απαραίτητο να έχουν αποθηκευτεί πριν αυτά δώσουν τα αντικείμενά τους στο Drawing περιβάλλον στο οποίο εν συνέχεια θα πρέπει να παραχθεί το Μηχανολογικό Σχέδιο των αντικειμένων αυτών. Για τον σκοπό αυτό δημιουργήθηκε η μέθοδος Close_And_Save_File() η οποία για ένα αρχείο παίρνει ως ορίσματα τα αντικείμενα Part, SheetMetalComponentDefinition, την τοποθεσία του «κρυφού» φάκελου Resources και το επιθυμητό όνομα αρχείου. Η υλοποίηση αυτής της μεθόδου σε επίπεδο εντολών παρουσιάζεται στο παρακάτω παράδειγμα.

```
Sub Close_And_Save_File(partDoc, partComp, ResourcesPath, mySaveName)

        If partComp.HasFlatPattern = True Then
            partComp.FlatPattern.ExitEdit()
        End If

        partDoc.SaveAs(ResourcesPath & mySaveName, True)
        partDoc.Close(True)

End Sub
```

Η μέθοδος αυτή αρχικά ελέγχει αν το αντικείμενο του αναπτύγματος έχει ξεχαστεί ενεργό σε κατάσταση επεξεργασίας (Edit). Σε τέτοια περίπτωση η κατάσταση αυτή απενεργοποιείται αφού αυτό το γεγονός δεν επιτρέπει την αποθήκευση ενός Part αρχείου (.ipt). Έπειτα το αρχείο αποθηκεύεται και στην συνέχεια κλείνει με σκοπό ένα Drawing αρχείο να ενεργοποιηθεί μελλοντικά.

Να γίνει σαφές ότι η εν λόγω μέθοδος ενεργεί για ένα αρχείο Part, ενός Κυλινδρικού Τμήματος. Επομένως για ένα δεύτερο αρχείο ενός άλλου Κυλινδρικού Τμήματος, η μέθοδος αυτή καλείται ακόμα μία φορά και σύμφωνα με τα αντικείμενα του αρχείου αυτού και το διαφορετικό επιθυμητό όνομα αποθήκευσής. Για παράδειγμα θα μπορεί να καλείται με τιμές εισόδου τα partComp2, partDoc2, mySaveName2.

Τέλος να αναφερθεί ότι μία ακόμα μέθοδος αποθήκευσης έχει δημιουργηθεί να αποθηκεύει στον φάκελο Resources, κρυφά από τον χρήστη, αχρεία κειμένου (.txt) τα οποία περιέχουν τους υπολογισμούς που έχουν πραγματοποιηθεί. Γενικότερα η αποθήκευση του οποιουδήποτε αρχείου στον φάκελο Resources είναι υψίστης σημασίας. Επομένως να αναφερθεί ότι η κάθε μέθοδος αποθήκευσης που έχει υλοποιηθεί, πρακτικά περιέχει περισσότερες εντολές που εξασφαλίζουν πάντα την σίγουρη αποθήκευση αυτών των αρχείων, με χρήση επαναληπτικών δομών και εντολών Try-Catch όπου επαναπροσδιορίζουν παράγοντες όπως το όνομα αποθήκευσης σε περίπτωση που κάτι πάει στραβά.

Στο σημείο αυτό έχει ολοκληρωθεί η περιγραφή της λειτουργείας του API στο Part περιβάλλον του Inventor και στην συνέχεια θα περιγραφεί η μέθοδος DWG(), η οποία έχει υλοποιηθεί να παράγει το Μηχανολογικό Σχέδιο εντός του περιβάλλοντος Drawing.

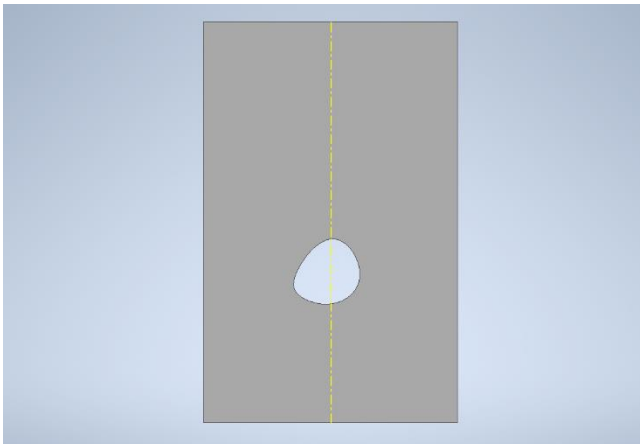
3.3.2 Μέθοδοι στο Drawing Περιβάλλον του Inventor

Στο παρόν υποκεφάλαιο θα περιγραφούν όλοι οι μέθοδοι που υλοποιήθηκαν να καλούνται και να επενεργούν εντός του περιβάλλοντος Inventor Drawing. Οι μέθοδοι αυτοί όπως και θα αναλυθεί στην συνέχεια, αποσκοπούν στην κατασκευή ενός μηχανολογικού σχεδίου που θα πληροί όλους τους απαραίτητους κανόνες. Για τον σκοπό αυτό υλοποιήθηκε η μέθοδος DWG() η οποία αντλεί πληροφορίες και διαχειρίζεται τα αντικείμενα που ήδη έχουν δημιουργηθεί εντός του περιβάλλοντος Inventor Part ενώ ταυτόχρονα επενεργεί εντός του περιβάλλοντος Drawing.

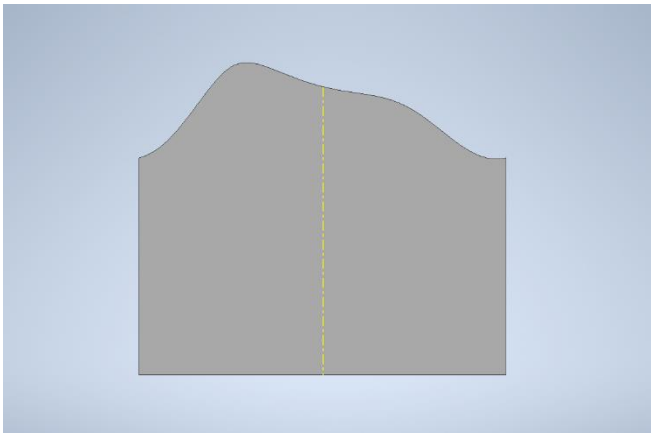
Μέθοδος DGW()

Η μέθοδος αυτή υλοποιήθηκε να ενεργεί για ένα Κυλινδρικό Τμήμα με σκοπό να παράγει το Μηχανολογικό Σχέδιο αυτού. Στις περιπτώσεις όπου δύο Κυλινδρικά Τμήματα περιέχονται, η μέθοδος αυτή επενεργεί μία φορά για το κάθε ένα και τοποθετεί τα απαραίτητα στοιχεία του, εντός του εικονικού φύλλου σχεδιάσεως (μεγέθους A3) που το περιβάλλον Drawing περιέχει. Επομένως δύο τροποποιημένοι μέθοδοι DWG() θα επενεργούν σε ένα πλήρες παραγόμενο Μηχανολογικό Σχέδιο του API όταν αυτό περιέχει δύο Κυλινδρικά Τμήματα. Η μέθοδος DWG() είναι υπέρογκη και προγραμματίστηκε να καλεί εντός της επιπλέον μεθόδους που υλοποιήθηκαν. Λόγο λοιπόν του υπέρογκου μεγέθους της, στην συνέχεια κυρίως θα γίνει περιγραφή της λειτουργείας της και αναφορά στις μεθόδους που καλεί.

Γενικότερα μία τέτοια μέθοδος αρχικά αντλεί το αντικείμενο του αναπτύγματος (Flatt Pattern) από ένα αρχείο Part (.ipt) και προβάλλει το περίγραμμά αυτού εντός ενός φύλου σχεδιάσεως. Ένα τέτοιο περίγραμμα αποφασιστικέ να οριοθετείτε εντός συγκεκριμένης περιοχής (10cm x 10cm) εντός του φύλου σχεδιάσεως, με σκοπό το Μηχανολογικό Σχέδιο να είναι ευπαρουσίαστο. Το γεγονός αυτό παρουσίασε δυσκολία αφού ένα ανάπτυγμα για τις όποιες διαστάσεις έχει δημιουργηθεί, μπορεί να είναι είτε υπερβολικά στενόμακρο κατά ύψος είτε κατά πλάτος. Ωστόσο οι καμπύλες των αναπτυγμάτων είναι αυτές που πρέπει να παρουσιάζονται απαραιτήτως αναλογικά και όχι τα εναπομείναντα πλαίσια των αναπτυγμάτων. Επομένως το κάθε ανάπτυγμα προκειμένου να παρουσιάζει την καμπύλη του έπρεπε να διακριθεί και να διαχειριστεί διαφορετικά. Έτσι ανάλογα με τον τύπο της καμπύλης που ένα ανάπτυγμα περιέχει αυτό διακρίθηκε σε δύο κατηγορίες, σύμφωνα με τις δύο αντίστοιχες φωτογραφίες που αποφαίνονται παρακάτω.



ΣΧΗΜΑ 3.5 – ΑΝΑΠΤΥΓΜΑ ΠΟΥ ΠΕΡΙΕΧΕΙ ΚΛΕΙΣΤΗ ΚΑΜΠΥΛΗ



ΣΧΗΜΑ 3.6 – ΑΝΑΠΤΥΓΜΑ ΠΟΥ ΠΕΡΙΕΧΕΙ ΚΑΜΠΥΛΗ ΠΑΡΑΛΛΗΛΗ ΣΤΗΝ ΒΑΣΗ ΤΟΥ

Στην συνέχεια λοιπόν θα περιγραφεί το πως η μέθοδος DWG() διαχειρίζεται την κάθε μία τέτοια περίπτωση αναπτυγμάτων, με σκοπό την οριοθετημένη μοντελοποίηση αυτών, εντός του εικονικού φύλου σχεδιάσεως του Drawing περιβάλλοντος.

Αναπτύγματα Που Περιέχουν Κλειστή Καμπύλη

Στην περίπτωση λοιπόν αναπτυγμάτων με κλειστή καμπύλη, υπολογίζονται αρχικά οι μέγιστες και ελάχιστες τιμές των X και Y της καμπύλης. Αυτό έγινε εφικτό με την υλοποίηση μιας επιπλέον τροποποιημένης μεθόδου Calculate(), η οποία δημιουργήθηκε να επιστρέφει τις μέγιστες και ελάχιστες τιμές μίας καμπύλης (minX, maxX, minY, maxY). Σύμφωνα με τις οριακές αυτές τιμές, υπολογίζονται το μέγιστο Ύψος και το μέγιστο Πλάτος μίας καμπύλης, σύμφωνα με τα οποία και η καμπύλη πρέπει να οροθετηθεί εντός συγκεκριμένης περιοχής στο φύλλο σχεδιάσεως.

- CurveHeight = maxY – minY
- CurveWidth = maxX - minX

Οι παραπάνω υπολογισμοί υλοποιήθηκαν εφόσον παρατηρήθηκε ότι η μέγιστη διάσταση (Ύψους ή Πλάτους) μίας τέτοιας καμπύλης είναι αυτή που πρέπει να καθορίζει τελικά και την κλίμακα (Scale) με την οποία θα προβάλλεται ένα ανάπτυγμα. Τελικά ο υπολογισμός της κλίμακας αποφασίστηκε να γίνεται σύμφωνα με το παρακάτω παράδειγμα υπολογισμού.

- Scale = Max(CurveHeight, CurveWidth) / 80

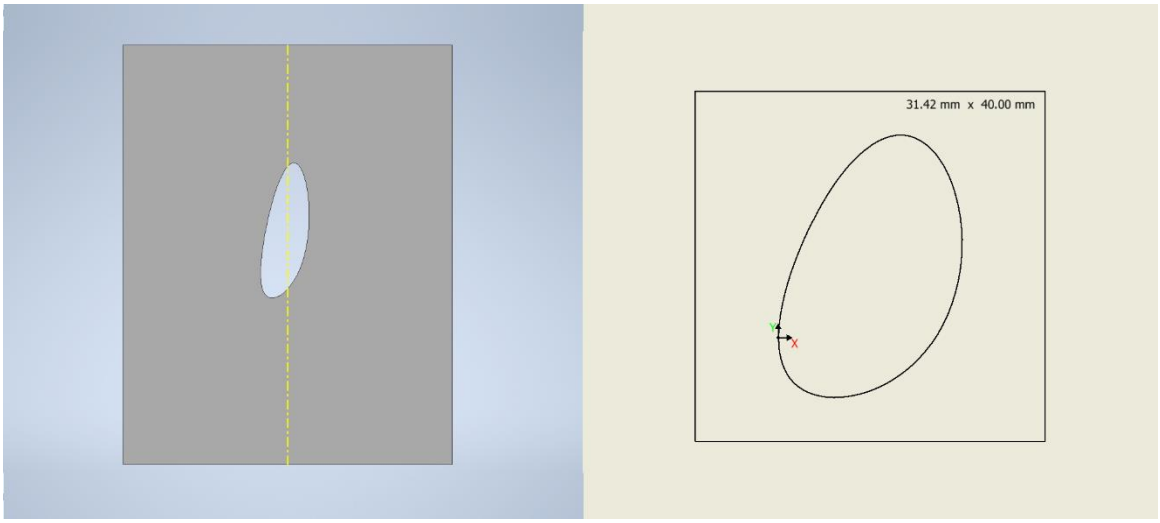
Σύμφωνα με τον υπολογισμό αυτής της κλίμακας, η σμίκρυνση (ή μεγέθυνση), του αναπτύγματος είναι τέτοια ώστε τελικά η καμπύλη να έχει 80mm μέγεθος στην μέγιστη διάστασή της. Έπειτα όλα τα υπόλοιπα μέρη (Edges) του αναπτύγματος γίνονται αόρατα και επόμενος μόνο η καμπύλη παραμένει στην οποία και δίνεται τοποθέτηση τέτοια ώστε αυτή να βρεθεί στο κέντρο ενός πλαισίου (100mm x 100mm) που ή ίδια μέθοδος σχεδιάζει.

Κάποιες φορές ανάλογα με την είσοδο που διέπει την καμπύλη, αυτή παρουσιάζεται ιδιαιτέρως στενόμακρη. Αυτό συμβαίνει όταν παρουσιάζει μεγάλη τιμή ο λόγος της μεγάλης της διάστασης σε σχέση με την μικρή της. Επομένως αυτό γίνεται αντιληπτό με την χρήση του ακόλουθου λόγου.

- Max(CurveHeight, CurveWidth) / Min(CurveHeight, CurveWidth)

Για παράδειγμα, όταν ο λόγος αυτός έχει ξεπεράσει το 10, αυτό σημαίνει ότι αφού η μεγάλη διάσταση έχει μέγεθος 80mm, η μικρή θα έχει λιγότερο από 0,8mm. Το γεγονός αυτό καθιστά μη ευανάγνωστη την απαρίθμηση σημείων εντός της καμπύλης στην αναλογική της κατάσταση και επομένως σε τέτοιες περιπτώσεις ένα ποιοτικό ανάπτυγμα παρουσιάζεται για λόγους βολικής προβολής και ανάγνωσης. Ωστόσο οι τιμές που παραθέτονται στους πίνακες είναι οι πραγματικές.

Τέτοιες μη βολικές καμπύλες παρουσιάζονται συνήθως όταν σαν είσοδο δοθεί μεγάλη γωνία στην 3η και 5η περίπτωση Κυλινδρικών Τομών που Υπολογίζονται. Τα ποιοτικά αναπτύγματα που παρουσιάζονται κατασκευάζονται εκ νέου σύμφωνα με τα ίδια δεδομένα εισόδου πλην της γωνίας όπου δίνεται με μικρότερη τιμή. Επομένως όλη η διαδικασία που περιγράφηκε εντός ενός περιβάλλοντος Part επανεκτελείται, προκειμένου να δημιουργηθεί ένα παρόμοιο ανάπτυγμα με παρόμοια κυρτότητα και σημεία καμπής. Ένα ποιοτικό ανάπτυγμα συνεπώς δεν κατασκευάζεται να είναι προκαθορισμένο. Κατασκευάζεται σύμφωνα με την είσοδο και με πειραγμένη την τιμή της γωνίας ώστε τελικά να περιέχει οπτικά την ρεαλιστική καμπύλη σε μία συμπιεσμένη εκδοχή της κάθετα στην στενόμακρη διάσταση της. Στην εικόνα που ακολουθεί αποφαίνεται ένα αρχικό ανάπτυγμα που περιέχει μία στενόμακρη καμπύλη, ενώ δίπλα του προβάλλεται το τελικό ανάπτυγμα που περιέχεται σε οριοθετημένο χώρο εντός του φύλου σχεδιάσεως του μηχανολογικού σχεδίου.



ΣΧΗΜΑ 3.7 – ΠΟΙΟΤΙΚΗ ΠΡΟΒΟΛΗ ΑΝΑΠΤΥΓΜΑΤΟΣ ΚΛΕΙΣΤΗΣ ΚΑΜΠΥΛΗΣ

Αναπτύγματα Με Καμπύλη Παράλληλη Στην Βάση Τους

Στην δεύτερη περίπτωση όπου ένα ανάπτυγμα διακρίνεται να περιέχει καμπύλη παράλληλη στην βάση του, η διαχείρισή ήταν διαφορετική. Επιλέχτηκε πάντα το πλάτος να λαμβάνει χώρο 100mm σύμφωνα με την παρακάτω εντολή.

- Scale = math.pi * Diameter / 100

Το πλάτος ενός τέτοιου αναπτύγματος αντιστοιχεί στην βάση ενός Κυλινδρικού Τμήματος και έχει πάντα τιμή (math.pi * Diameter). Με την χρήση λοιπόν αυτής της κλίμακας (Scale), η σμίκρυνση (ή μεγέθυνση) του αναπτύγματος είναι τέτοια ώστε τελικά η βάση του αναπτύγματος να έχει 100mm μέγεθος εντός του φύλλου σχεδιάσεως. Επιλέχτηκε η βάση να καταλαμβάνει σταθερό μέγεθος αφού σε αυτήν την κατηγορία αναπτυγμάτων πάντα η καμπύλη εξελίσσετε παράλληλα με την βάση.

Πλέον η διάσταση του ύψους παραμένει πρόβλημα οι οποία πρέπει ομοίως να περιοριστεί εντός το πολύ 100mm. Επομένως υπολογίζονται οι μέγιστες και οι ελάχιστες τιμές του ύψους της καμπύλης. Ομοίως λοιπόν με πριν μια επιπλέον τροποποιημένη μέθοδος Calculate() δημιουργήθηκε να επιστρέφει τις τιμές minY και maxY της καμπύλης, μέσω των οποίων υπολογίζεται το Ύψος που η καμπύλη περιέχει.

- CurveHeight = maxY – minY

Γνωρίζουμε επιπλέον ότι το πλάτος μίας τέτοιας καμπύλής είναι ίσο με την βάση του αναπτύγματος το οποίο έχει τοποθετηθεί να λαμβάνει μήκος 100mm στο φύλο σχεδιάσεως. Συνεπώς γνωρίζουμε ότι:

- CurveWidth = Βάση Ολόκληρου Του Αναπτύγματος = math.pi * διάμετρος = 100mm

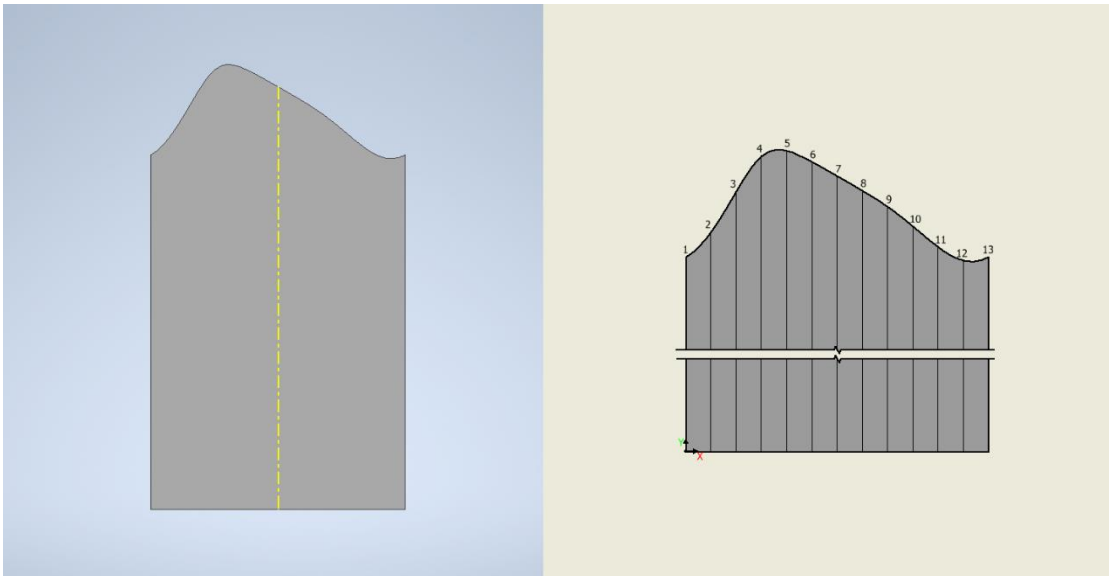
Σε ένα ανάπτυγμα τέτοιου τύπου δεν γίνονται αόρατα τα υπόλοιπα μέρη του, αλλά σκοπός είναι αυτό να παρουσιάζεται ολόκληρο, μιας και η καμπύλη του κινείται παράλληλα στην βάση του. Συνεπώς αποφάσεις λαμβάνονται προκειμένου διάσταση του ύψους του συνολικού αναπτύγματος να περιοριστεί εντός το πολύ 100mm όταν αυτό τα ξεπερνάει. Αντίστοιχα όταν το συνολικό ύψος είναι πολύ μικρό, για λόγους βολικής προβολής και ανάγνωσης πρέπει το ανάπτυγμα να εμφανίζεται ψηλότερο.

Στα αναπτύγματα αυτού του τύπου, κάτω από την καμπύλη, εξελίσσεται το πλαισίου του αναπτύγματος μέχρι και την βάση του. Αυτό δεν αποτελεί σημαντική πληροφορία και επομένως μπορεί να αυξομειωθεί δίχως πρόβλημα.

Σύμφωνα λοιπόν με τα παραπάνω και με σκοπό την οριοθετημένη προβολή, υλοποιήθηκε για τα αναπτύγματα της εν λόγω κατηγορίας τέσσερίς διαφορετικές αποφάσεις να μπορούν να ληφθούν. Οι αποφάσεις αυτές λαμβάνονται ανάλογα με την σχέση του συνολικού ύψους του αναπτύγματος (maxY), σε σχέση με πλάτος του (math.pi * διάμετρος), το οποίο όπως έχει αναφέρει έχει οροθετηθεί να καταλαμβάνει μήκος 100mm εντός του Σχεδίου. Στην συνέχεια λοιπόν ακολουθεί περιγραφή των ενεργειών που πραγματοποιούνται από την μέθοδο DWG(), σε κάθε μία περίπτωση.

1) Όταν maxY <= math.pi * διάμετρος ΚΑΙ maxY > = math.pi * διάμετρος / 2 , η προβολή του αναπτύγματος γίνεται ως αυτό έχει κατασκεπαστεί σύμφωνα με την είσοδο του χρήστη αφού το συνολικό ύψος του στο σχέδιο θα καταλαμβάνει μήκος από 50mm έως και το πολύ 100mm, γεγονός που καθιστά ένα τέτοιο ανάπτυγμα ευπαρουσίαστο.

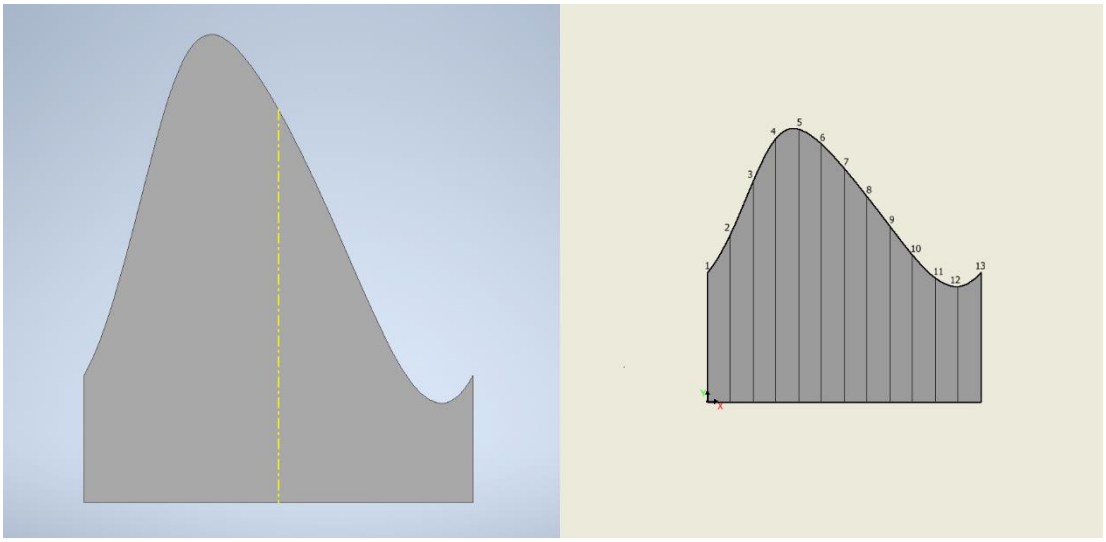
2) Όταν maxY > math.pi * διάμετρος το ανάπτυγμα δεν μπορεί να καταλαμβάνει οριοθετημένο χώρο (100mm x100mm), και έτσι παρουσιάζεται με μειωμένο ύψος στο πλαίσιο κάτω της καμπύλης. Όταν αυτό συμβαίνει καλείται η έτοιμη μέθοδος Break() η οποία έχει την δυνατότητα να κόβει ένα κομμάτι από το πλαίσιο ενός αναπτύγματος. Σύμφωνα λοιπόν με την εικόνα που ακολουθεί το ανάπτυγμα τελικά παρουσιάζεται να περιέχει μία ρωγμή που σημαίνει ότι το πλαίσιό του αναλογικά δεν καταλαμβάνει το ύψος που του παρουσιάζεται, ενώ ωστόσο η καμπύλη έχει παραμείνει αναλλοίωτη και παρουσιάζεται αναλογικά κλιμακωμένη.



ΣΧΗΜΑ 3.8 – ΠΡΟΒΟΛΗ ΣΤΕΝΟΜΑΚΡΟΥ ΑΝΑΠΤΥΓΜΑΤΟΣ ΜΕ ΚΑΜΠΥΛΗ ΠΑΡΑΛΛΗΛΗ ΣΤΗΝ ΒΑΣΗ ΤΟΥ

3) Όταν maxY < math.pi*Διάμετρος /2 το ανάπτυγμα δεν μπορεί να είναι ευανάγνωστο λόγο του κοντού του ύψους και επομένως όλη η διαδικασία της τρισδιάστατης κατασκευής που περιγράφηκε εντός ενός περιβάλλοντος Part επαναλαμβάνετε, προκειμένου να δημιουργηθεί ένα νέο όμοιο ανάπτυγμα, με μεγαλύτερο Ύψος. Έτσι η μέθοδος Model_3D_Construction() καλείτε πάλι με τα ίδια ορίσματα πλην του ορίσματος του ύψους του κυλίνδρου, με σκοπό αυτό να έχει μεγαλύτερη τιμή τέτοια ώστε τελικά, maxY = CurveWidth, προκειμένου το ανάπτυγμα να μπορεί να οροθετηθεί σε τετράγωνη περιοχή (100mm x 100mm). Επόμενος το νέο αυτό ανάπτυγμα παρουσιάζεται εντός του μηχανολογικού σχεδίου στο οποίο ομοίως με πριν η μέθοδος Break() επενεργεί με σκοπό την τοποθέτηση της αντίστοιχης ρωγμής στο πλαίσιο του αναπτύγματος. Στην περίπτωση αυτή, ομοίως πάλι με το προηγούμενο παράδειγμα η καμπύλη έχει παραμείνει αναλλοίωτη και παρουσιάζεται αναλογικά κλιμακωμένη.

4) Στην περίπτωση όπου το ύψος της καμπύλης είναι από μόνο του πολύ μεγάλο παρουσιάζεται ένα ποιοτικό ανάπτυγμα. Όταν δηλαδή το ύψος της καμπύλης από μόνο του υπερβαίνει το συνολικό πλάτος του αναπτύγματος, τότε δεν αρκεί ακόμα και αν κοπεί όλο το μέρος του εναπομείναντος πλαισίου του αναπτύγματος που περιέχεται κάτω από την καμπύλη. Έχοντας λοιπόν οριοθετήσει το πλάτος στα 100mm, όταν το ύψος της καμπύλης είναι αρκετό από μόνο του για να το ξεπεράσει τότε αναγκαστικά ένα ποιοτικό ανάπτυγμα δημιουργείται. Τέτοιες μη βολικές καμπύλες παρουσιάζονται συνήθως όταν σαν είσοδο δοθεί μεγάλη γωνία στην 1η, 3η και 5η περίπτωση Κυλινδρικών Τομών που Υπολογίζονται. Τα ποιοτικά αναπτύγματα που παρουσιάζονται τελικά στο μηχανολογικό σχέδιο κατασκευάζονται εκ νέου σύμφωνα με τα ίδια δεδομένα εισόδου πλην της γωνίας όπου δίνεται με μικρότερη τιμή. Επομένως όλη η διαδικασία που περιγράφηκε εντός ενός περιβάλλοντος Part επανεκτελείται, προκειμένου να δημιουργηθεί ένα παρόμοιο ανάπτυγμα με παρόμοια κυρτότητα και σημεία καμπής. Ένα ποιοτικό ανάπτυγμα συνεπώς δεν είναι προκαθορισμένα διαφορετικό αλλά κατασκευάζεται ώστε να περιέχει οπτικά την ρεαλιστική καμπύλη σε μία συμπιεσμένη εκδοχή της κάθετα στην στενόμακρη διάστασή της. Στην εικόνα που ακολουθεί αποφαίνεται το οριοθετημένο μηχανολογικό σχέδιο ενός τέτοιου αναπτύγματος σε σχέση με το αρχικό ανάπτυγμα που έχει δημιουργηθεί σύμφωνα με τις τιμές εισόδου.



ΣΧΗΜΑ 3.9 – ΠΟΙΟΤΙΚΗ ΠΡΟΒΟΛΗ ΑΝΑΠΤΥΓΜΑΤΟΣ ΜΕ ΚΑΜΠΥΛΗ ΠΑΡΑΛΛΗΛΗ ΣΤΗΝ ΒΑΣΗ ΤΟΥ

Άλλες Μέθοδοι που καλούνται εντός της μεθόδου DWG()

Πέραν της αυστηρής οριοθέτησης των αναπτυγμάτων, η μέθοδος DWG() είναι υπέρογκη και υλοποιήθηκε να καλεί υπεράριθμες ακόμα μεθόδους. Οι μέθοδοι που η DWG() καλεί, υλοποιήθηκαν και επενεργούν εντός του Drawing περιβάλλοντος με σκοπό την αυτοματοποιημένη παραγωγή του μηχανολογικού σχεδίου των εν λόγω αναπτυγμάτων και σύμφωνα με όλους τους απαραίτητους κανόνες που το διέπουν. Στην συνέχεια λοιπόν ακολουθεί η αναφορά και συνοπτική περιγραφή των εν λόγω μεθόδων:

- Η μέθοδος Axis_System() υλοποιήθηκε να κατασκευάζει και να τοποθετεί το σύστημα αξόνων σε σημείο του αναπτύγματος που έχει θεωρηθεί ως κέντρο.
- Η μέθοδος Matrix() υλοποιήθηκε να κατασκευάζει και να συμπληρώνει τον πίνακα που περιέχει τις υπολογισμένες τιμές (X,Y) των δειγματοληπτισμένων σημείων.
- Η μέθοδος Model_3D() υλοποιήθηκε να τοποθετεί στο Μηχανολογικό Σχέδιο τα μοντέλα των Τρισδιάστατων Κυλινδρικών Τμημάτων, οριοθετημένα με παρόμοιο τρόπο, τέτοιο ώστε η τρισδιάστατη τομή να αποφαίνεται πάντα ολόκληρη.
- Η μέθοδος Show_Dimensions() υλοποιήθηκε να προβάλει τις διαστάσεις των ακμών των πλαισίων των αναπτυγμάτων και των τρισδιάστατων μοντέλων.
- Η μέθοδος Show_Points() υλοποιήθηκε να απαριθμεί τα σημεία μίας καμπύλής καθώς και να δημιουργεί της ακμές της δειγματοληψίας. Η μέθοδος αυτή παρουσίασε μεγάλη δυσκολία καθώς έπρεπε να γίνει κατάλληλη επιλογή και τοποθέτηση σημείων έτσι ώστε αυτά να μην εμφανίζονται απανωγραμμένα μεταξύ τους και ούτε να είναι τοποθετημένα πάνω στην καμπύλη. Αυτό εξαρτάται από την θέση του κάθε σημείου σε σχέση με την ίδια την καμπύλη και τα διπλανά του σημεία. Επόμενος ο ρυθμός μεταβολής της καμπύλης για ένα σημείο ήταν η κύρια συνιστώσα που συντέλεσε στην λύση του προβλήματος.
- Γενικότερα επιπλέον εντολές υλοποιήθηκε να εμφανίζουν κατάλληλα μνήματα όπου και όποτε αυτά είναι απαραίτητα εντός του φύλλου σχεδιάσεως.
- Τέλος υλοποιήθηκαν, να κατασκευάζεται το υπόμνημα στην κάτω δεξιά γωνία και επιπλέον να παρουσιάζονται στην πάνω δεξιά γωνία τα δεδομένα εισόδου καθώς και μία προκαθορισμένη εικόνα που περιέχει τις διαστάσεις των δεδομένων αυτών. Για τον εν λόγω σκοπό υλοποιήθηκε και επενεργεί η μέθοδος Ypomnima().

3.4 Έλεγχος των Τιμών Εισόδου

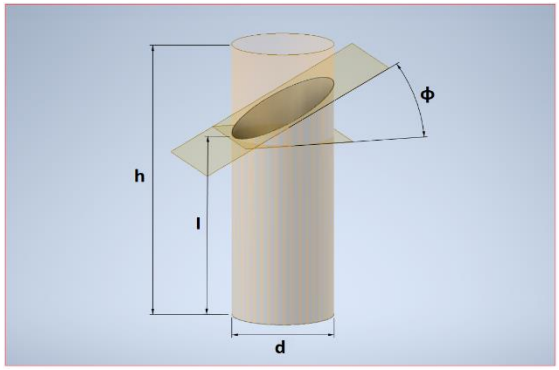
Η κατασκευή των Μοντέλων εξαρτάται άμεσα από τις τιμές εισόδου και δεν είναι πάντα εφικτή για τις οποιεσδήποτε τιμές. Προκειμένου λοιπόν το API να εκτελείται δίχως σφάλματα, τα δεδομένα εισόδου ελέγχονται πριν από την εκτέλεσή του. Σε περίπτωση που οι είσοδοι είναι λανθασμένοι κατάλληλα διαγνωστικά μηνύματα παρουσιάζονται στην ενεργή φόρμα που ο χρήστης αλληλεπιδρά, ενώ η εκτέλεση του API δεν πραγματοποιείται. Στο παρόν υπο-κεφάλαιο θα γίνει η περιγραφή του ελέγχου των τιμών εισόδου για την κάθε μία εκ των πέντε περιπτώσεων Κυλινδρικών Τομών.

Αρχικά αποφασιστικέ και οι πέντε περιπτώσεις να δέχονται τιμές μήκους μόνο από 1 mm έως 9999mm. Επιπλέον αποφασίστηκε να λαμβάνονται υπόψιν μόνο δύο δεκαδικά ψηφία για κάθε τιμή εισόδου. Έπειτα αποφασίστηκε ο έλεγχος των τιμών εισόδου της κάθε περίπτωσης να επιτυγχάνεται με έλεγχο ανισώσεων, πολλές εκ των οποίων και αποδείχτηκαν με χρήση τριγωνομετρίας.

Στην συνέχεια λοιπόν, για την κάθε μία εκ των πέντε περιπτώσεων Κυλινδρικών Τομών, παρατίθενται όλες οι απαραίτητες πλάγιες όψεις με την βοήθεια των οποίων αποφασίστηκαν – αποδείχτηκαν οι απαραίτητες ανισώσεις που ελέγχουν το πεδίο τιμών της εισόδου της κάθε περίπτωσης.

Τομή Κυλίνδρου Με Πλάγιο Επίπεδο

Στην περίπτωση του μοντέλου αυτού, οι ανισώσεις που αποφαίνονται παρακάτω δεξιά, αποφασίστηκε να αποτελούν των έλεγχο των αποδεκτών τιμών εισόδου του. Για την κατανόησή των εν λόγω ανισώσεων, απαραίτητη είναι η εικόνα πλάγιας όψης του εν λόγω μοντέλου, η οποία και παρατίθεται στα αριστερά.

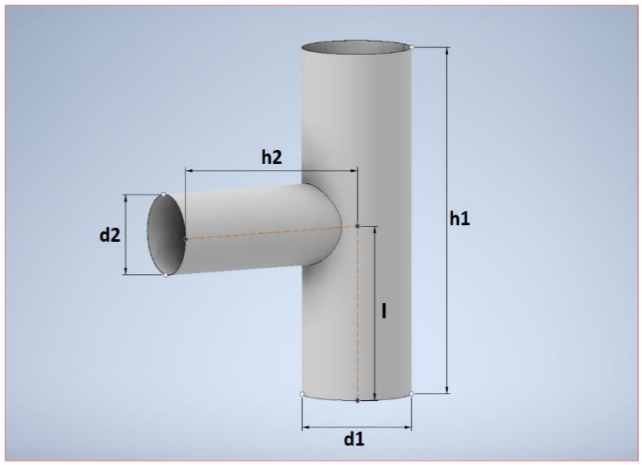


ΣΧΗΜΑ 3.10 – ΠΛΑΓΙΑ ΌΨΗ 1ΗΣ ΠΕΡΙΠΤΩΣΗΣ ΚΥΛΙΝΔΡΙΚΩΝ ΤΟΜΩΝ

- **$0 < \phi < 90$**
- **$l > 0$**
- **$h > l + d * \tan(\phi)$**

Τομή Κυλίνδρου με Κάθετο Κύλινδρο

Παρομοίως και για την περίπτωση του εν λόγω μοντέλου, οι ανισώσεις που αποφαίνονται παρακάτω δεξιά, αποφασίστηκε να αποτελούν των αποδεκτών τιμών εισόδου του. Ομοίως απαραίτητη είναι η εικόνα πλάγιας όψης του, η οποία και παρατίθεται στα αριστερά.

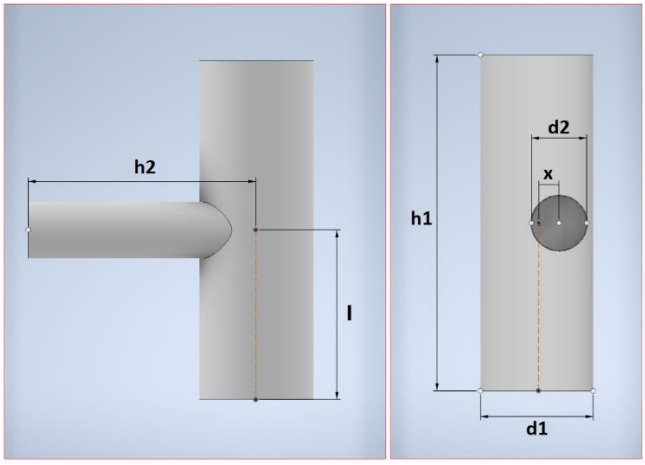


ΣΧΗΜΑ 3.11 – ΠΛΑΓΙΑ ΌΨΗ 2ΗΣ ΠΕΡΙΠΤΩΣΗΣ ΚΥΛΙΝΔΡΙΚΩΝ ΤΟΜΩΝ

- **d1 >= d2**
- **l > d2/2**
- **h1 > l + d2/2**
- **h2 > d1/2**

Τομή Κάθετων Κυλίνδρων Με Απόσταση Ανάμεσα Στους Άξονες

Ομοίως και για την περίπτωση αυτή, σύμφωνα με την εικόνα πλάγιας όψης που ακολουθεί, αποφασίστηκε ότι οι εν λόγω διαστάσεις πρέπει να επαληθευθούν τις ακόλουθες ανισώσεις. Στην περίπτωση αυτή η απόσταση ανάμεσα στους άξονες (x) αποτελεί νέα συνιστώσα της οποίας το αποδεκτό πεδίο τιμών εισόδου οροθετήθηκε.

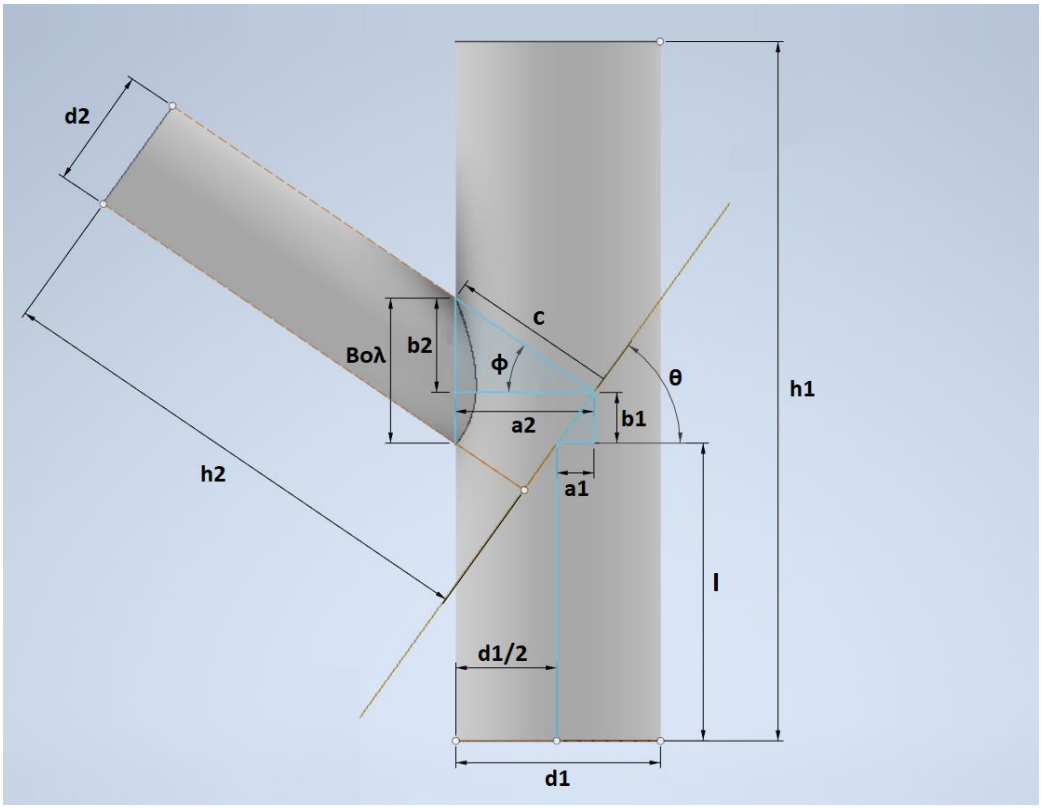


ΣΧΗΜΑ 3.12 – ΠΛΑΓΙΑ ΌΨΗ 3ΗΣ ΠΕΡΙΠΤΩΣΗΣ ΚΥΛΙΝΔΡΙΚΩΝ ΤΟΜΩΝ

- **d1 > d2**
- **l > d2/2**
- **h1 > l + d2/2**
- **h2 > d1/2**
- **0 < x < (d1-d2)/2**

Τομή Κυλίνδρων Υπό Κλίση

Το εν λόγω μοντέλο εξαρτάται πλέων άμεσα και από την τιμή της γωνίας (φ), πέραν υπολοίπων τιμών των κυλινδρικών διαστάσεων (h1, h2, d1, d2, l). Οι ανισώσεις λοιπόν που αποφασίστηκε να ελέγχουν την είσοδο του εν λόγω μοντέλο αποδείχτηκαν με χρήση τριγωνομετρίας, κατά την οποία τρίγωνα αναλύθηκαν εντός της πλάγιας όψης του εν λόγω μοντέλου. Κάτωθι λοιπόν παρατίθενται η εν λόγω πλάγια όψη όπου ήταν απαραίτητη κατά την απόδειξη των ανισώσεων αυτών.



ΣΧΗΜΑ 3.13 – ΠΛΑΓΙΑ ΌΨΗ 4ΗΣ ΠΕΡΙΠΤΩΣΗΣ ΚΥΛΙΝΔΡΙΚΩΝ ΤΟΜΩΝ

Σύμφωνα λοιπόν με τις διαστάσεις που αποφαίνονται στην ανωτέρω εικόνα πλάγιας όψης, αρχικά αποδείχτηκαν οι εξισώσεις ακολουθούν στα αριστερά, με την βοήθεια των οποίων τελικά αποδείχτηκαν οι επιθυμητές ανισώσεις που ακολουθούν στα δεξιά.

- **θ = 90 – ϕ**
- **b1 = sin(θ) * d2/2**
- **a2 = cos(θ) * d2/2**
- **b2 = tan(ϕ)**
- **c = $\sqrt{a2^2 + b2^2}$**
- **Βολ = b1 + b2**

- **0 < ϕ < 90**
- **d1 >= d2**
- **l > d2/2**
- **h1 > l + Βολ**
- **h2 > c**

Τομή Κυλίνδρων Υπό Κλίση Και Με απόσταση Ανάμεσα Στους Άξονες

Η περίπτωση αυτή αποτελεί των συνδυασμό της 3η και 4ης περίπτωσης. Δηλαδή τόσο η συνιστώσα της μεταβλητής γωνίας όσο και η συνιστώσα της μεταβλητής απόστασης ανάμεσα στους άξονες συνυπάρχουν. Συνεπώς ήταν δύσκολη η μαθηματική απόδειξη των ανισώσεων που ελέγχουν τις τιμές εισόδου οι οποίες καθιστούν εφικτή την ύπαρξη ενός τέτοιου μοντέλου. Παρατηρήθηκε ωστόσο ότι εφόσον ικανοποιούνται οι ανισότητες των προηγούμενων περιπτώσεων (**3ης** και **4ης**), ένα τέτοιο μοντέλο θα ήταν πάντα εφικτό να δημιουργηθεί με ασφάλεια. Αυτό παρατηρήθηκε να ισχύει αφού για μηδενική απόσταση ανάμεσα στους άξονες (x=0 , 4η περίπτωση), παρουσιάζονται οι μέγιστες οριακές τιμές για τα ασφαλή ύψη των δύο κυλινδρικών τμημάτων (minH1,minH2). Συνεπώς θα είναι ασφαλής η κατασκευή ενός μοντέλου της παρούσας περίπτωσης όταν οι τιμές ύψους των Κυλίνδρων επαληθεύουν τις συνθήκες της **4ης περίπτωσης**.

Επομένως σε πρώτο βήμα υλοποιήθηκε να δημιουργείτε ένα μοντέλο όταν ικανοποιούνται οι ακόλουθες ανισώσεις, γνωρίζοντας ότι αυτό θα κατασκευαστεί με ασφάλεια.

- **d1 > d2**
- **l > d2/2**
- **h1 > l + Βολ**
- **h2 > c**
- **0 < x < (d1-d2)/2**

Ωστόσο όταν οι ανωτέρω ανισώσεις δεν ικανοποιούνται, θα είναι ακόμα εφικτό για ελαφρός μικρότερες τιμές h1 και h2 να δημιουργηθεί ένα μοντέλο και επομένως οι τιμές αυτές έπρεπε να εξεταστούν. Επομένως υλοποιήθηκε πάλι η μαθηματική ισχύς του Inventor να τις υπολογίσει για εμάς. Έτσι έπειτα από την μη ικανοποίηση των προηγούμενων συνθηκών, ο χρόνος εκτέλεσης αυξάνεται μιας και επιπλέον κώδικας υλοποιήθηκε να κατασκευάζει ασφαλή μοντέλα πάνω στα οποία υπολογίζονται και επιστρέφονται οι ελάχιστες δυνατές τιμές ύψους h1 και h2.

Επομένως στην διαδικασία αυτή αρχικά θέτονται ως είσοδο οι ασφαλείς τιμές του ύψους των κυλινδρικών τμημάτων, σύμφωνα με τα παρακάτω:

- **h1 = l + Βολ**
- **h2 = c**

Για τις ανωτέρω τιμές λοιπόν και σε συνδυασμό με τις υπόλοιπες δεδομένες τιμές από τον χρήστη (d1, d2, I, φ, x), αρχικά κατασκευάζεται ένα αντίστοιχο και σίγουρα ασφαλές μοντέλο. Έπειτα χρησιμοποιώντας το μοντέλο αυτό, μία νέα μέθοδος ParamCalculate() υπολογίζει και επιστρέφει τις ελάχιστες επιτρεπτές τιμές ύψους των κυλινδρικών τμημάτων (minH1, minH2).

Εφόσον λοιπόν οι αρχικές τιμές εισόδου (h1, h2) είναι μεγαλύτερες από τις ελάχιστα επιτρεπτές τιμές που υπολογίστηκαν, τότε τελικά το API ενεργοποιείται και επενεργεί σύμφωνα με τις εισόδους που αρχικά ζητήθηκαν. Στην περίπτωση που οι αρχικές τιμές εισόδου (h1, h2) δεν είναι μεγαλύτερες από τις οριακές τιμές που υπολογίστηκαν, αντίστοιχα μνήματα σφάλματος παρουσιάζονται στον χρήστη και τον ενημερώνουν για τις ελάχιστες επιτρεπτές τιμές. Έτσι ο χρήστης θα έχει γνώση των εν λόγω ελάχιστων τιμών (minH1, minH2), με σκοπό την ορθή επανυποβολή τους, διατηρώντας ωστόσο ίδιες τις υπόλοιπες δεδομένες τιμές (d1, d2, I, φ, x).

Στην περίπτωση που ικανοποιούνται οι αρχικές ανισώσεις, τότε δεν λαμβάνει τόπο η διαδικασία που περιγράφηκε. Όταν σε πρώτο στάδιο, οι προηγούμενες ανισώσεις υποδεικνύουν την ασφαλή κατασκευή των μοντέλων, τότε δεν υπάρχει λόγος για πρόσθετο χρόνο εκτέλεσης και αναμονής από τον χρήστη (~10 sec).

Γενικότερα σε κάθε περίπτωση, ο έλεγχος των τιμών εισόδου ήταν ζωτικής σημασίας εφόσον προλαμβάνει την πιθανή αδυναμία εκτέλεσης της εφαρμογής. Σε περίπτωση που το Inventor API κληθεί να κατασκευάσει ένα μοντέλο που δεν ικανοποιεί τις ανισώσεις που αποδείχτηκαν, σφάλματα θα προκύψουν, και θα οδηγήσουν την εφαρμογή σε αντικανονικό τερματισμό (Crash). Για τον ίδιο λόγο επίσης αποφασιστικέ να οροθετηθεί το εύρος όλων των εισόδων μήκους (1mm - 9999mm), καθώς η μεγάλη διαφορά στην τάξη μεγέθους των τιμών αυτών θα μπορούσε να δημιουργήσει αδυναμία εκτέλεσης της εφαρμογής. Αυτό αφού το Inventor αδυνατεί να διαχειριστεί ταυτόχρονα στο ίδιο αρχείο τιμές με μεγάλη διαφορά στην τάξη μεγέθους, όπως για παράδειγμα τιμές $d1=10^{20}$ mm και d2=1mm.

Κλείνοντας λοιπόν το παρόν υποκεφάλαιο, μπορούμε να συνοψίσουμε στο ότι όλοι οι απαραίτητοι έλεγχοι πραγματοποιούνται με κύριο σκοπό την εξασφάλιση της ορθής εκτέλεσης της εφαρμογής για κάθε περίπτωση εισόδου, και επομένως η σημασία τους είναι ζωτική.

Κεφάλαιο 4

Αποτελέσματα

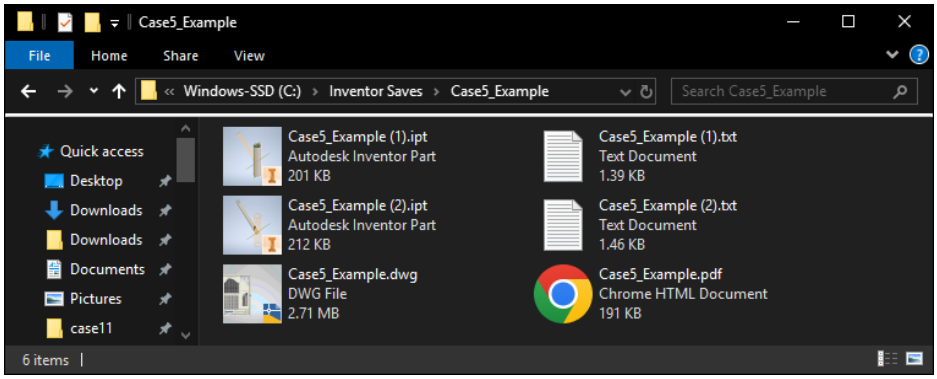
Όπως έχει αναφερθεί η τελική υλοποίηση δεν είναι άλλο από μία ολοκληρωμένη εφαρμογή Inventor VBA. Η εφαρμογή αυτή περιέχει μια γραφική διεπαφή χρήστη με σκοπό για κάποιες περιπτώσεις Κυλινδρικών Τομών, να παράγει τα τρισδιάστατα μοντέλα, να υπολογίζει τα δισδιάστατα αναπτύγματα αυτών, και να παράγει το μηχανολογικό σχέδιο των δισδιάστατων αναπτυγμάτων. Σημαντικό επίσης είναι το γεγονός ότι η λειτουργεία της γραφικής διεπαφής χρήστη υλοποιήθηκε να ελέγχει τις τιμές εισόδου και να εμφανίζει διαγνωστικά μνήματα σφάλματος. Στο παρόν κεφάλαιο λοιπόν θα γίνει η περιγραφή και η παράθεση των τελικών αποτελεσμάτων – παραγομένων της εφαρμογής αυτής, όπως και θα παρουσιαστούν οι λειτουργείες της γραφικής διεπαφής χρήστη σε περίπτωση ορθής αλλά και σε εσφαλμένης υποβολής δεδομένων.

4.1 Περιγραφή του Τελικού Αποτελέσματος

Η εφαρμογή που υλοποιήθηκε λοιπόν προγραμματιστικέ να περιέχει μία γραφική διεπαφή χρήστη, να ελέγχει τις τιμές εισόδου και να καλεί ρουτίνες εντός της εφαρμογής Autodesk Inventor με σκοπό την παραγωγή αρχείων Inventor Part (.ipt) και Inventor Drawing (.dwg), όπως και αρχείων PDF και αρχείων κειμένου (.txt).

Έπειτα λοιπόν από την ορθή υποβολή των δεδομένων και την ολοκληρωμένη εκτέλεση του Inventor API, τα παραγόμενα αυτά αρχεία θα έχουν δημιουργηθεί και θα είναι δυνατή η αποθήκευσή τους. Τα παραγόμενα αρχεία Inventor Part θα περιέχουν το τρισδιάστατο μοντέλο ενός κυλινδρικού τμήματος, και την υλοποιημένη δισδιάστατη ανάπτυξη αυτού. Τα αρχεία Inventor Drawing θα περιέχουν ένα μηχανολογικό σχέδιο αναπτυγμάτων το οποίο θα πληροί όλους τους κανόνες μηχανολογικού σχεδίου. Τα αρχεία PDF θα περιέχουν την προβολή του Μηχανολογικού Σχεδίου που θα έχει δημιουργηθεί και θα περιέχεται στα αρχεία (.dwg). Τέλος τα αρχεία κειμένου θα περιέχουν το επιθυμητό πλήθος υπολογισμένων σημείων ενός αναπτύγματος, του όπιου αντίστοιχου κυλινδρικού τμήματος.

Δύο κυλινδρικά τμήματα περιέχονται στις περιπτώσεις 2,3,4 και 5. Επομένως στις περιπτώσεις αυτές, για κάθε κυλινδρικό τμήμα παράγεται και από ένα αντίστοιχο αρχείο (.ipt) και (.txt). Κατά την αποθήκευση ο χαρακτηρισμός (1) αφορά πάντα το κάθετο κυλινδρικό τμήμα όπως μπορεί να φανεί στην παρακάτω εικόνα. Στην εικόνα λοιπόν που ακολουθεί αποφαίνονται εντός φακέλου τα τελικά παραγόμενα που έχουν παραχθεί και αποθηκευτεί έπειτα από μια εκτέλεση της εφαρμογής που υλοποιήθηκε.



ΣΧΗΜΑ 4.1 – ΣΥΝΟΛΙΚΑ ΠΑΡΑΓΟΜΕΝΑ ΑΡΧΕΙΑ ΕΝΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΕΝΤΟΣ ΦΑΚΕΛΟΥ

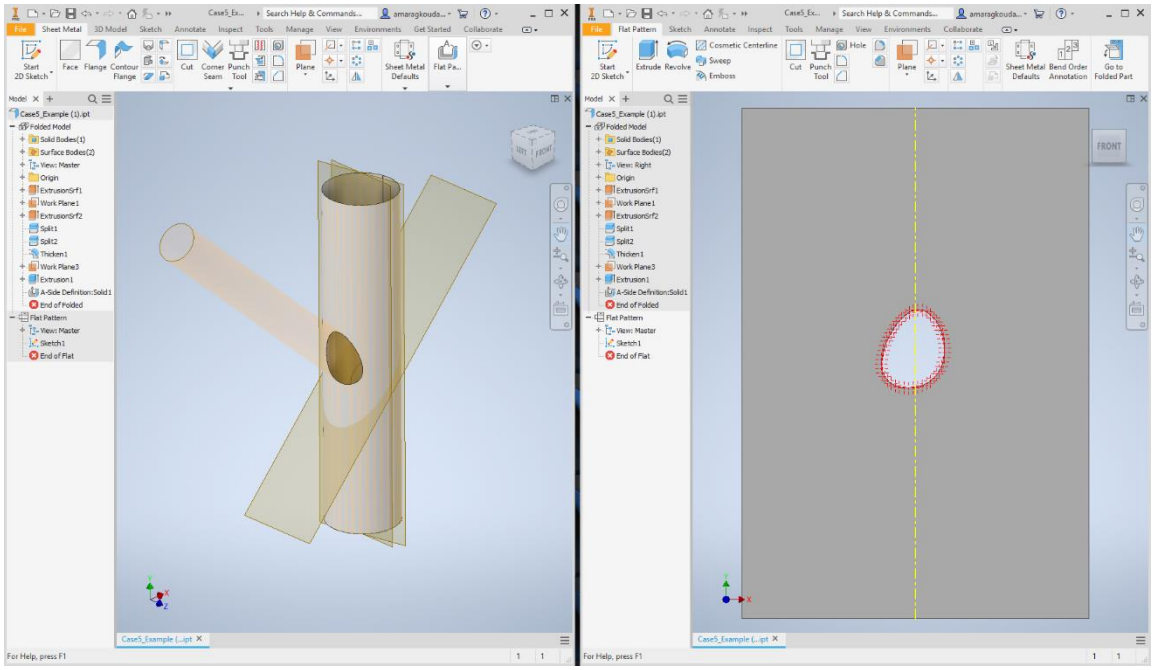
4.2 Παράθεση Παραγομένων

Στο παρόν υποκεφάλαιο θα γίνει η παράθεση των παραγόμενων αρχείων που αναφέρθηκαν προηγουμένως. Θα παρουσιαστούν τα παραγόμενα αρχεία μόνο της 5ης περίπτωσης. Δεν υπάρχει λόγος παράθεσης των αποτελεσμάτων των υπόλοιπων περιπτώσεων καθώς η 5η περίπτωση αποτελεί την πολυπλοκότερη όλων.

Αποτελέσματα Παραγόμενων Αρχείων Inventor Part (.ipt)

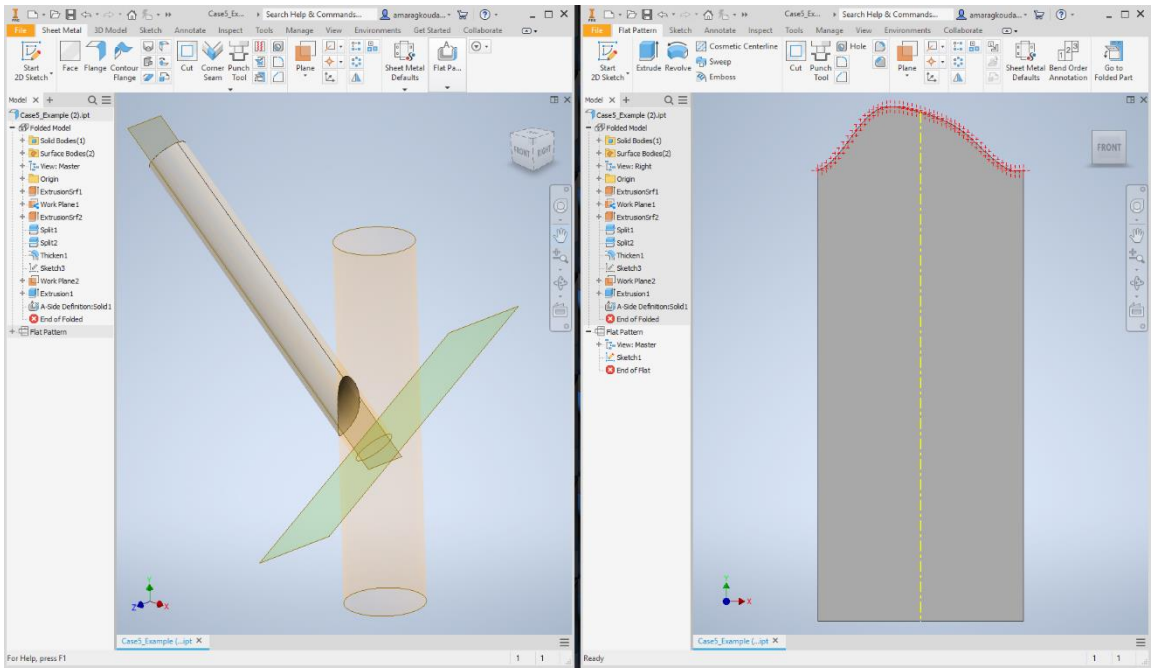
Τα αρχεία αυτά παράγονται να περιέχουν το τρισδιάστατο μοντέλο ενός κυλινδρικού τμήματος, και την υλοποιημένη δισδιάστατη ανάπτυξη αυτού. Τα αρχεία αυτά είναι σημαντικό να αποθηκεύονται σε περίπτωση που ο χρήστης γνωρίζει την λειτουργεία του Inventor CAD και επιθυμεί να τροποποιήσει τα τρισδιάστατα μοντέλα που αυτά περιέχουν.

Στην εικόνα που ακολουθεί λοιπόν, παρουσιάζεται το αρχείο Case5_Example(1).ipt το οποίο αφορά ένα κάθετο κυλινδρικό τμήμα της 5ης περίπτωσης υπολογισμού. Στην εν λόγω εικόνα στα αριστερά το αρχείο αυτό περιέχει εντός του Inventor το τρισδιάστατο μοντέλο, ενώ στα δεξιά το ίδιο αρχείο περιέχει το ανάπτυγμα αυτού (Flatt Pattern).



ΣΧΗΜΑ 4.2 – ΠΑΡΑΓΟΜΕΝΟ ΑΡΧΕΙΟ ΤΥΠΟΥ PART ΕΝΟΣ ΚΑΤΑΚΟΡΥΦΟΥ ΚΥΛΙΝΔΡΙΚΟΥ ΤΜΗΜΑΤΟΣ

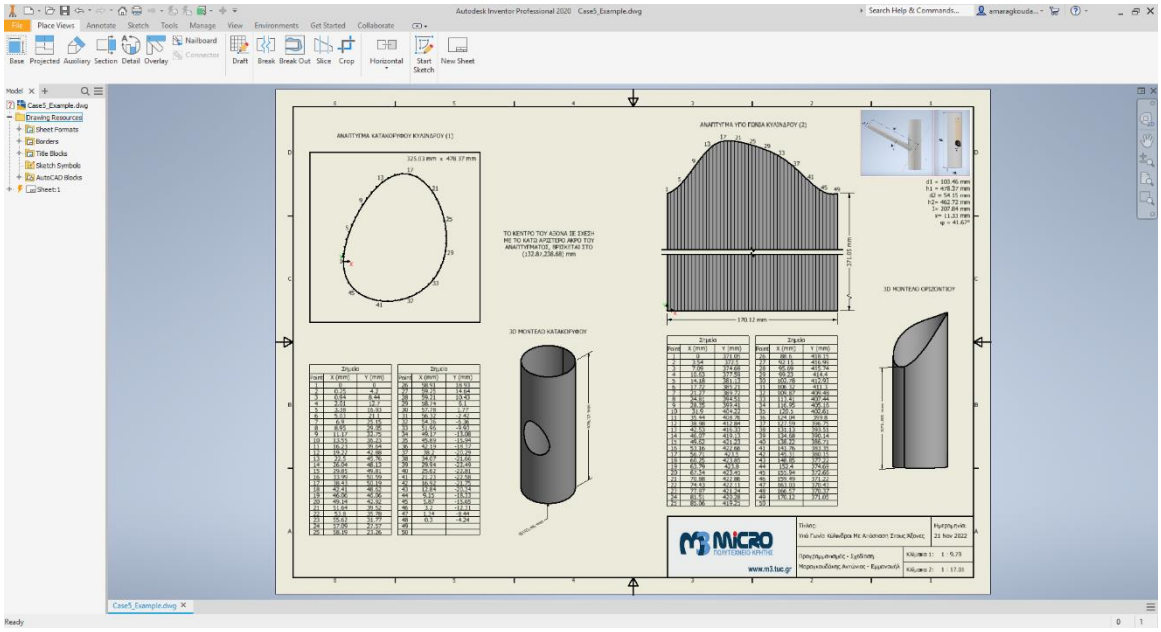
Ομοίως στην εικόνα που ακολουθεί παρουσιάζεται το αρχείο Case5_Example(2).ipt το οποίο αφορά το υπό γωνία κυλινδρικό τμήμα της ίδιας περίπτωσης και εκτέλεσης. Στα αριστερά της εικόνας αυτής, το εν λόγω Inventor Part αρχείο περιέχει το αντίστοιχο τρισδιάστατο μοντέλο, ενώ στα δεξιά περιέχει το αντίστοιχο ανάπτυγμα αυτού.



ΣΧΗΜΑ 4.3 – ΠΑΡΑΓΟΜΕΝΟ ΑΡΧΕΙΟ ΤΥΠΟΥ PART ΕΝΟΣ ΥΠΟ ΓΩΝΙΑ ΚΥΛΙΝΔΡΙΚΟΥ ΤΜΗΜΑΤΟΣ

Αποτελέσματα Παραγόμενων Αρχείων Inventor Drawing (.dwg)

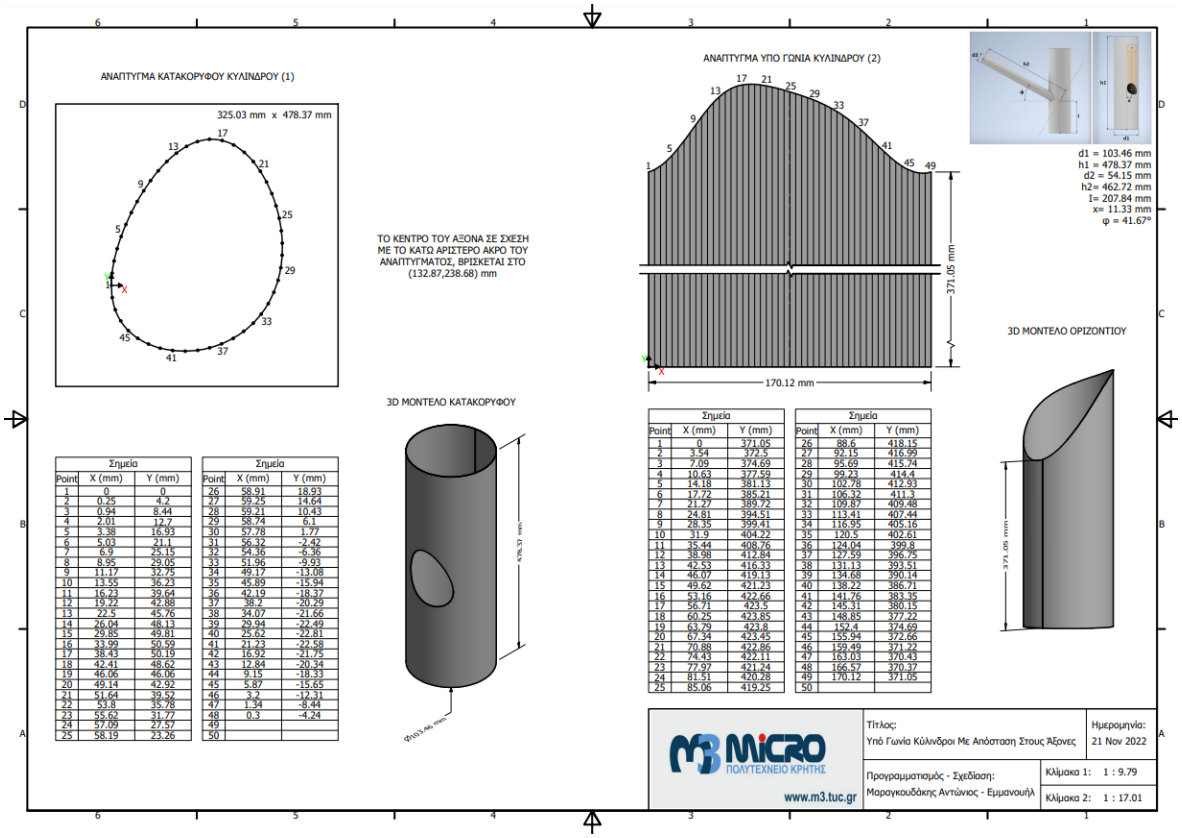
Τα αρχεία αυτά παράγονται να περιέχουν συνολικά για ένα ή δύο κυλινδρικά τμήματα, το μηχανολογικό σχέδιο των δισδιάστατων αναπτυγμάτων. Τα αρχεία αυτά είναι σημαντικό να αποθηκεύονται σε περίπτωση που ο χρήστης γνωρίζει την λειτουργεία του Inventor CAD και επιθυμεί να τροποποιήσει τα περιεχόμενα του μηχανολογικού σχεδίου που αυτά περιέχουν. Στην εικόνα που ακολουθεί λοιπόν παρουσιάζεται εντός του Inventor Drawing, το παραγόμενο αρχείο Case5_Example.dwg, το οποίο περιέχει το Μηχανολογικό Σχέδιο των αναπτυγμάτων των κυλινδρικών τμημάτων που παρουσιάστηκαν προηγουμένως.



ΣΧΗΜΑ 4.4 – ΠΑΡΑΓΟΜΕΝΟ ΑΡΧΕΙΟ ΤΥΠΟΥ DRAWING ΕΝΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ 5ΗΣ ΠΕΡΙΠΤΩΣΗΣ

Αποτελέσματα Παραγόμενων Αρχείων PDF

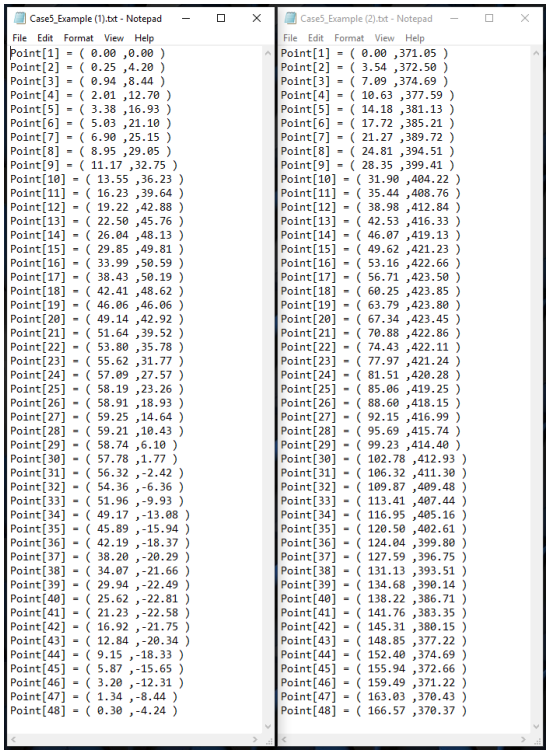
Τα αρχεία αυτά παράγονται να περιέχουν την προβολή του Μηχανολογικού Σχεδίου που έχει δημιουργηθεί και περιέχεται στα αρχεία Drawing (.dwg). Τα αρχεία αυτά είναι σημαντικό να αποθηκεύονται σε περίπτωση που ο χρήστης δεν επιθυμεί την αλληλεπίδραση με την εφαρμογή του Inventor. Ένα αρχείο PDF είναι χρήσιμο για την άμεση προβολή και εκτύπωση ενός μηχανολογικού σχεδίου που πρόκειται να μελετηθεί. Στην εικόνα του ακολουθεί λοιπόν αποφαίνεται το περιεχόμενο του αρχείου Case5_Example.pdf.



ΣΧΗΜΑ 4.5 – ΠΑΡΑΓΟΜΕΝΟ ΑΡΧΕΙΟ PDF ΕΝΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ 5ΗΣ ΠΕΡΙΠΤΩΣΗΣ

Αποτελέσματα Παραγόμενων Αρχείων Κειμένου (.txt)

Τα αρχεία αυτά παράγονται να περιέχουν τους υπολογισμούς των σημείων του αναπτύγματος ενός κυλινδρικού τμήματος. Τα αρχεία αυτά είναι σημαντικό να αποθηκεύονται αφού οι τιμές των υπολογισμών που περιέχουν μπορεί να φανούν πολύ χρήσιμες σε περίπτωση που μία τρίτη εφαρμογή δημιουργηθεί με σκοπό να τις επεξεργάζεται ή να της δρομολογεί σε ένα μηχάνημα δισδιάστατης κοπής. Στην εικόνα του ακολουθεί αποφαίνεται το περιεχόμενο των αρχείων Case5_Example(1).txt και Case5_Example(2).txt. Στην εν λόγω εικόνα λοιπόν, το αρχείο στα αριστερά περιέχει τους υπολογισμούς του αναπτύγματος του κατακόρυφου κυλινδρικού τμήματος, ενώ το αρχείο στα δεξιά περιέχει τους υπολογισμούς του αναπτύγματος του υπό γωνία κυλινδρικού τμήματος.



ΣΧΗΜΑ 4.6 – ΠΑΡΑΓΟΜΕΝΑ ΑΡΧΕΙΑ TXT ΕΝΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ 5ΗΣ ΠΕΡΙΠΤΩΣΗΣ

Κλείνοντας, σημαντικό είναι πάλι να αναφερθεί το γεγονός ότι τα αρχεία μηχανολογικού σχεδίου (.dwg) και (.pdf), δεν παράγονται για επιλογή υπολογισμού πέραν των 50 σημείων. Αυτό αποφασίστηκε να συμβαίνει διότι για μεγάλο αριθμό υπολογισμών δημιουργείται πρόβλημα δυσανάγνωστης προβολής του Μηχανολογικού Σχεδίου. Ωστόσο σε περίπτωση που ο χρήστης δεν ενδιαφέρεται για το Μηχανολογικό Σχέδιο, αλλά ενδιαφέρεται για πολλούς υπολογισμούς σημείων, τα αρχεία (.ipt) και (.txt) υλοποιήθηκε να δημιουργούνται για υπολογισμούς έως και 10000 σημείων.

4.3 Αποτελέσματα Γραφικής Διεπαφής Χρήστη

Όπως έχει αναφερθεί κύριο μέρος της εργασίας αποτέλεσε η υλοποίηση μία γραφικής διεπαφής χρήστη η οποία προγραμματιστικέ να επιτελεί τρείς βασικές λειτουργίες. Πρώτη λειτουργεία της προγραμματίστηκε να είναι η αρχική επιλογή μίας περίπτωσης εκ των πέντε περιπτώσεων Κυλινδρικών Τομών που υλοποιήθηκε να υπολογίζονται. Επόμενη βασική λειτουργεία της πραγματοποιήθηκε να είναι η παρουσίαση διαγνωστικών μνημάτων σφάλματος έπειτα από την ανίχνευση της μη έγκυρης

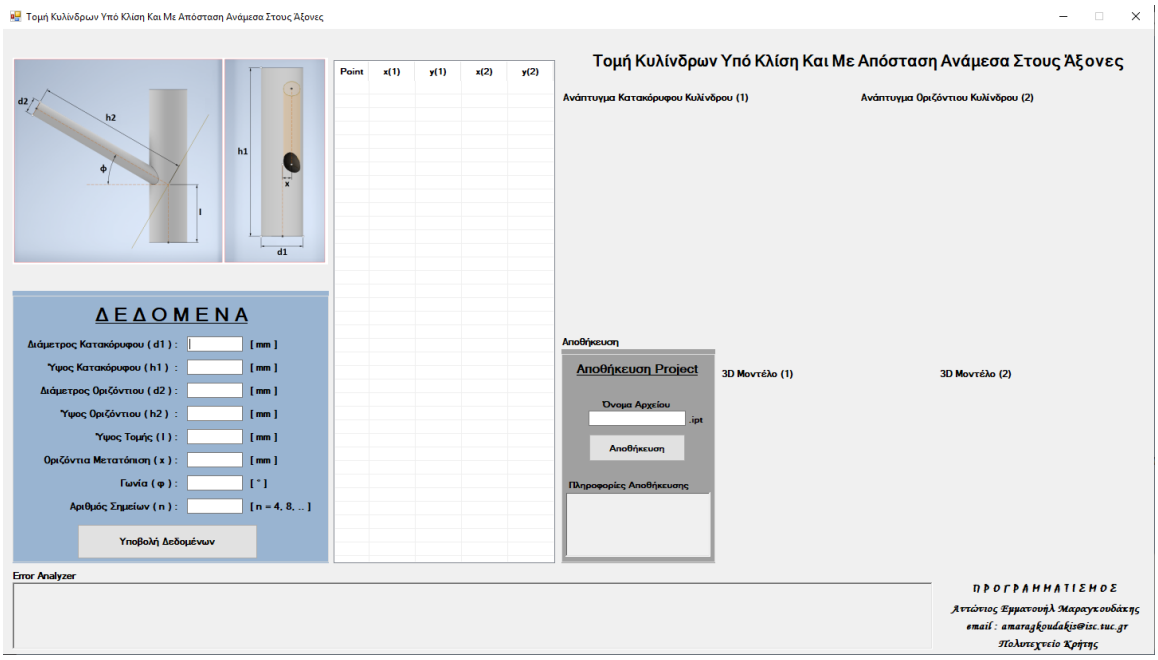
υποβολής δεδομένων. Τελευταία λειτουργεία της γραφικής διεπαφής χρήστη υλοποιήθηκε να είναι η παρουσίαση υπολογισμών και εικόνων, όπως και η δυνατότητα της αποθήκευσης των τελικών παραγομένων. Στο παρόν υποκεφάλαιο λοιπόν θα περιγραφεί η συμπεριφορά που υλοποιήθηκε να παρουσιάζει η γραφική διεπαφή χρήστη κατά την διάρκεια εκτέλεσης της εφαρμογής.

Όπως έχει αναφερθεί λοιπόν, κατά την εκκίνηση της εφαρμογής, η γραφική διεπαφή χρήστη υλοποιήθηκε μόνο παρέχει την δυνατότητα της επιλογής του υπολογισμού μίας περίπτωσης κυλινδρικών τομών. Στην εικόνα που ακολουθεί αποφαίνεται η αρχική φόρμα που προβάλλεται στον χρήστη, κατά την εκκίνηση της εφαρμογής.

ΣΧΗΜΑ 4.7 – Η ΕΦΑΡΜΟΓΗ ΚΑΤΑ ΤΗΝ ΕΚΚΙΝΗΣΗ ΤΗΣ

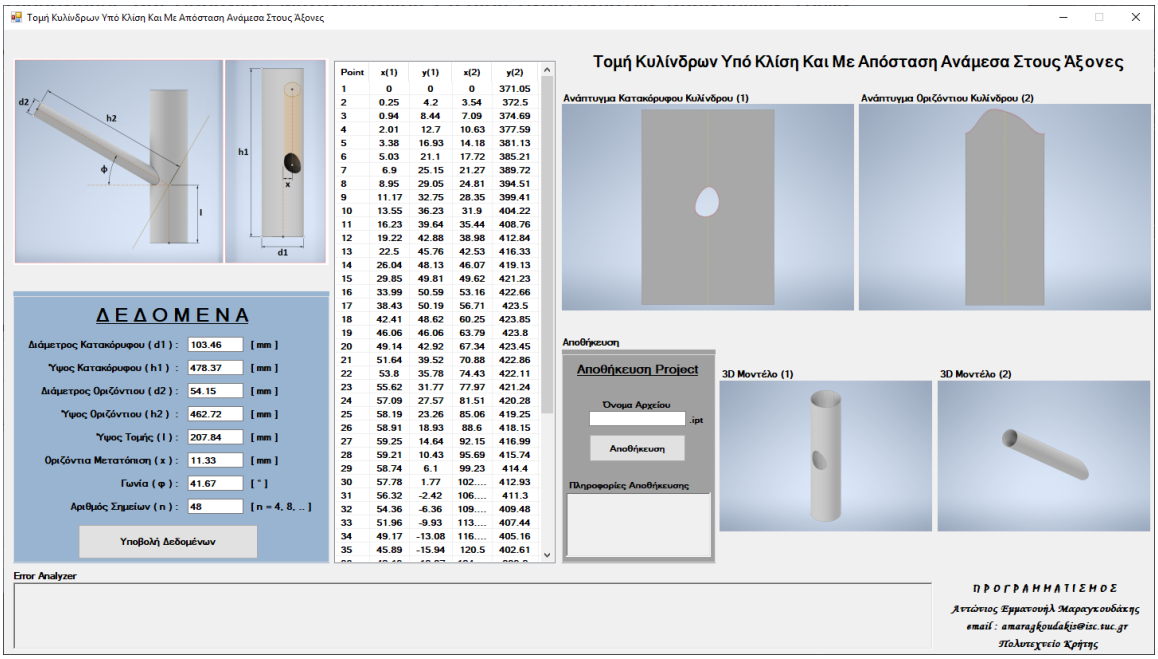
Στο σημείο αυτό, να αναφερθεί ότι στην φόρμα αυτή επίσης υλοποιήθηκαν και οι δύο εικόνες της κάτω αριστερής γωνίας να είναι αντικείμενα Button. Η λειτουργεία που τους προσδόθηκε είναι να αναδύουν το προεπιλεγμένο πρόγραμμα διαδικτυακής περιήγησης, εντός του οποίου και θα προβάλλονται οι αντίστοιχες αρχικές ιστοσελίδες του Πολυτεχνείου Κρήτης καθώς και του εργαστηρίου M3 Micro.

Γενικότερα, όσων αφορά τις υπόλοιπες λειτουργείες της εν λόγω φόρμας, έπειτα από την επιλογή μίας περίπτωσης κυλινδρικών τομών (Button - Εικόνα), μία αντίστοιχη φόρμα προβάλλεται η οποία έχει ως σκοπό και επιτελεί όλες τις λειτουργείες που έχουν περιγραφεί στο υποκεφάλαιο 3.2.1. Στην εικόνα λοιπόν που ακολουθεί παρουσιάζεται η φόρμα που προβάλετε όταν για παράδειγμα έχει επιλεγεί 5η περίπτωση υπολογισμού.



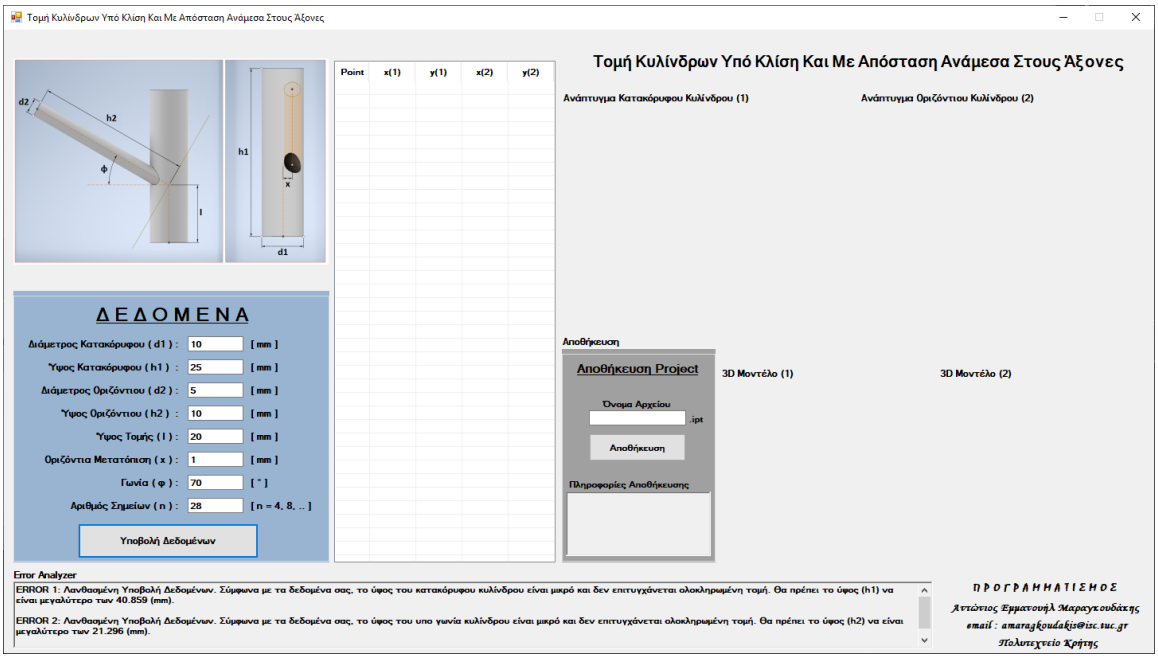
ΣΧΗΜΑ 4.8 – Η ΕΦΑΡΜΟΓΗ ΚΑΤΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΗΣ 5ΗΣ ΠΕΡΙΠΤΩΣΗΣ

Έπειτα από τη ορθή υποβολή δεδομένων, ο κώδικας του API θα αρχίσει να επενεργεί και να παράγει τα επιθυμητά αντικείμενα εντός του Autodesk Inventor. Όταν η εκτέλεσή του θα έχει ολοκληρωθεί, επιθυμητές πληροφορίες θα έχουν παρουσιαστεί εντός της ανωτέρω φόρμας. Στην εικόνα που ακολουθεί λοιπόν παρουσιάζεται η εν λόγω φόρμα έπειτα από την υποβολή ορθών δεδομένων και την εκτέλεση του API.



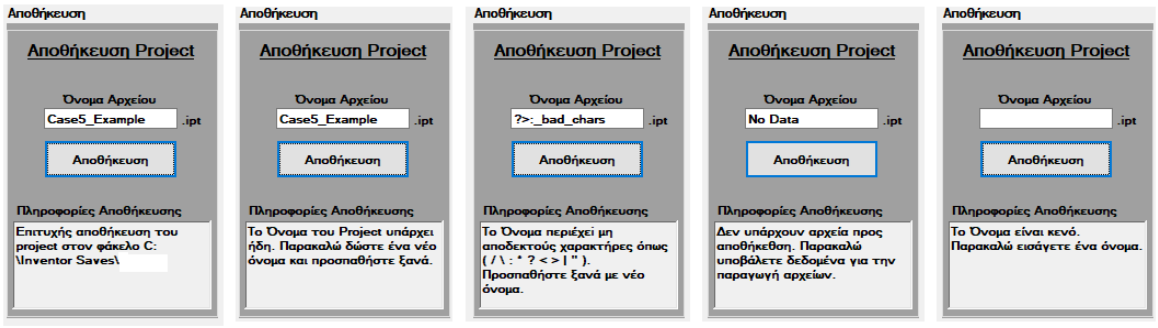
ΣΧΗΜΑ 4.9 – Η ΕΦΑΡΜΟΓΗ ΈΠΕΙΤΑ ΑΠΟ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΗΣ 5ΗΣ ΠΕΡΙΠΤΩΣΗΣ

Όταν τα δεδομένα που έχουν υποβληθεί δεν είναι έγκυρα, η εκτέλεση του Inventor API δεν λαμβάνει τόπο. Στην περίπτωση αυτή κατάλληλα διαγνωστικά μηνύματα παρουσιάζονται στην φόρμα που ο χρήστης αλληλεπιδρά. Στην εικόνα που ακολουθεί λοιπόν παρουσιάζεται η λειτουργεία της διάγνωσης σφαλμάτων της ανωτέρω φόρμας, έπειτα από την υποβολή και ανίχνευση μη έγκυρων τιμών εισόδου.



ΣΧΗΜΑ 4.10 – Η ΕΦΑΡΜΟΓΗ ΈΠΕΙΤΑ ΑΠΟ ΤΗΝ ΔΙΑΓΝΩΣΗ ΣΦΑΛΜΑΤΟΣ ΤΩΝ ΤΙΜΩΝ ΕΙΣΟΔΟΥ

Κλείνοντας, σημαντικό είναι επίσης να αναφερθεί το ότι παρομοίως έχει υλοποιηθεί η λειτουργεία της διάγνωσης σφαλμάτων και κατά την διαδικασία της αποθήκευσης των παραγομένων. Στην εικόνα που ακολουθεί λοιπόν παρουσιάζονται οι πέντε πιθανές περιπτώσεις διάγνωσης κατά την λειτουργεία της αποθήκευσης των παραγομένων.



ΣΧΗΜΑ 4.11 – ΔΙΑΓΝΩΣΕΙΣ ΣΦΑΛΜΑΤΟΣ ΚΑΤΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΠΑΡΑΓΟΜΕΝΩΝ

Κεφάλαιο 5

Συμπεράσματα

Έχοντας ολοκληρωθεί η εκπόνηση της εν λόγω διπλωματικής εργασίας, στο κεφάλαιο αυτό θα περιγραφούν τα συμπεράσματα και γενικότερα οι δυνατότητες του Inventor API. Επιπλέον θα αναφερθούν οι περιορισμοί και οι δυσκολίες που προέκυψαν κατά την διαδικασία της υλοποίησης, καθώς και θα γίνει λόγος για πιθανές μελλοντικές εργασίες που είναι δυνατόν να επακολουθήσουν της εργασίας αυτής.

5.1 Συμπεράσματα – Δυνατότητες Inventor API

Οι δυνατότητες μίας εφαρμογής Inventor API είναι τεράστιες στον τομέα της αυτοματοποίησης διαδικασιών εντός ενός περιβάλλοντος CAD. Με την χρήση κατάλληλων εντολών είναι δυνατή η αυτοματοποιημένη παραγωγή αρχείων Part, Assembly, Drawing και Presentation.

Πέραν αυτού υψίστης χρησιμότητας αποτελεί το γεγονός ότι ένα Inventor API μπορεί να επενεργεί με σκοπό όχι μόνο την παραγωγή αρχείων, αλλά και με σκοπό των υπολογισμό επιθυμητών τιμών, αφού οι ρουτίνες που μπορεί να καλούνται εντός του μπορούν να τιθασεύουν την τρισδιάστατη μαθηματική ισχύ του Autodesk Inventor. Πέραν αυτού, ένα Inventor API μπορεί επιπλέον να επιτρέπει τον υπολογισμό βέλτιστων μοντέλων, αφού είναι δυνατόν επαναληπτικά να δημιουργούνται τροποποιημένα μοντέλα, τα οποία και να συγκρίνονται με σκοπό την εύρεση του βέλτιστου. Οι δυνατότητες λοιπόν ενός Inventor API είναι απεριόριστες τόσο από μηχανολογικής όσο και από μαθηματικής απόψεως.

Από την σκοπιά της πληροφορικής, ένα Inventor API διέπεται από ιεραρχία αντικειμένων και υλοποιείται με τη χρήση αντικειμενοστραφή προγραμματισμού. Το γεγονός αυτό καθιστά βατή την κατανόηση και την γραφή των εντολών, ωστόσο είναι μεγάλος ο όγκος προετοιμασίας και εργασίας ενός Inventor API, λόγο του υπέρογκου μοντέλου ιεραρχίας αντικειμένων που το διέπει.

Ωστόσο, γενικότερα όσον αφορά την υλοποίηση ενός Inventor API, ο προγραμματισμός δεν φάνηκε να είναι το μόνο απαραίτητο. Πέραν της γνώσης προγραμματισμού, απαραίτητη προϋπόθεση για την δημιουργία ενός Inventor API φαίνεται να αποτελεί η άριστη γνώση της χρήσης του Inventor, η τρισδιάστατη γεωμετρική κατανόηση και η γνώση μαθηματικών.

5.2 Δυσκολίες και Περιορισμοί

Η συνολική εργασία μπορεί να διαχωριστεί σε τρία βασικά μέρη. Πρώτο μέρος αποτελεί ή δημιουργία και ο προγραμματισμός των λειτουργειών της γραφικής διεπαφής χρήστη που περιγράφηκε. Δεύτερο μέρος αποτελεί ο προγραμματισμός του Inventor API, ενώ τρίτο μέρος αποτελεί ο έλεγχος των τιμών εισόδου τόσο δε επίπεδο θεωρητικής ανάλυσης όσο και σε επίπεδο υλοποίησης. Στην συνέχεια λοιπόν ακολουθεί η περιγραφή των δυσκολιών και των περιορισμών που παρατηρήθηκαν στο κάθε ένα μέρος της εργασίας.

Δυσκολίες και περιορισμοί κατά την ανάπτυξη της Γραφικής Διεπαφής Χρήστη

Όσον αφορά το πρώτο μέρος δεν παρουσιάστηκαν ιδιαίτερες δυσκολίες. Η χρήση των Windows Forms είναι ιδιαιτέρως βολική καθώς το βασικό στήσιμο των γραφικών στοιχείων είναι εφικτό με χρήση μεταφοράς και απώθησης (drag and drop). Ωστόσο απαραίτητη ήταν η γνώση της λειτουργείας του κάθε γραφικού στοιχείου, προκειμένου να είναι εφικτός ο συνολικός προγραμματισμός της διεπαφής, ο οποίος και διεκπεραιώθει με τη χρήση αντικειμενοστραφή προγραμματισμού.

Δυσκολίες και περιορισμοί κατά την ανάπτυξη του Inventor API

Όσον αφορά το δεύτερο μέρος της εργασίας, ο προγραμματισμός του Inventor API απαίτησε την άριστη κατανόηση και χρήση της εφαρμογής Autodesk Inventor. Επιπλέον λόγο του ότι το Inventor API διέπεται από ένα υπέρογκο μοντέλο ιεραρχίας αντικειμένων, αντιστοίχως υπέρογκη ήταν και η προεργασία με σκοπό την κατανόησή του μοντέλου αυτού. Πέραν τούτου κατά την διαδικασία του προγραμματισμού του API, ιδιαίτερη δυσκολία παρουσίασε το γεγονός ότι οι μέθοδοι που υποστηρίζονται αποδέχονται ορίσματα διάφορων τύπων, που συχνά μπορούσαν να περιέχουν τιμές

εκτός ορίων με αποτέλεσμα την αντικανονική διακοπή της εκτέλεσης (Crash). Όταν αυτό συνέβαινε δεν υπήρχε σαφή επεξήγηση για το ποιο όρισμα περιείχε τιμή εκτός ορίων, με αποτέλεσμα να δυσχεραίνεται η διεκπεραίωση της εκσφαλμάτωσης (Debugging). Επόμενος η διεκπεραίωση της εκσφαλμάτωσης συνεχώς απαιτούσε την τρισδιάστατη γεωμετρική κατανόηση και τον υπολογιστικό έλεγχο του κάθε αντικειμένου, με σκοπό την εύρεση αυτού που περιείχε τιμές εκτός ορίων.

Οι περιορισμοί που παρατηρήθηκε να υπάρχουν σε επίπεδο προγραμματισμού του Inventor API είναι ίδιοι με αυτούς που ομοίως παρατηρούνται και σε επίπεδο χρήσης της εφαρμογής. Ένας περιορισμός παρατηρήθηκε να είναι το γεγονός ότι το Inventor δεν μπορεί να διαχειρίζεται ταυτόχρονα στο ίδιο αρχείο τιμές μικρής τάξης μεγέθους, με τιμές μεγάλης τάξης μεγέθους. Επιπλέον περιορισμός παρατηρήθηκε το γεγονός ότι για την ανάπτυξη ενός κυλινδρικού μοντέλου, ήταν απαραίτητη η δημιουργία μίας σχισμής σε αυτό προκειμένου το μοντέλο να είναι δυνατόν να ξεδιπλωθεί στον δυσδιάστατο χώρο. Οι δύο αυτοί περιορισμοί σε συνδυασμό αποτέλεσαν μικρό εμπόδιο σύμφωνα με την περιγραφή που ακολουθεί.

Το γεγονός ότι η σχισμή δεν είναι δυνατόν να έχει μηδενική τιμή, οδήγησε στην απόφαση η τιμή αυτή να δίνεται όσον το δυνατότερο μικρότερη. Η σχισμή αυτή αποφασιστικέ να φέρει όσον το δυνατόν μικρότερη τιμή εφόσον ανάλογα με το μέγεθός της, επηρεάζει την ακρίβεια και την αρτιότητα των παραγόμενων αναπτυγμάτων. Το γεγονός αυτό συχνά οδηγούσε σε σφάλμα σε συνδυασμό με τον πρώτο περιορισμό. Αυτό το σφάλμα συνέβαινε λόγο της διαφοράς της τάξη μεγέθους της μικρής σχισμής σε σχέση με την τάξη μεγέθους των υπολοίπων διαστάσεων.

Για παράδειγμα έχοντας διάμετρο 100000mm, το μήκος της βάσης ενός αναπτύγματος περιορίζεται να μην είναι ακριβώς π*D = 314159,265… , αλλά λιγότερο, αναλογικά με το μέγεθος της σχισμής. Ουσιαστικά ο περιορισμός αυτός περιορίζει μόνο την ανάγκη μεγάλης ακρίβειας μιας θεωρητικής ανάλυσης και αποφασιστικέ η σχισμή να φέρει όσον το δυνατών μικρότερη τιμή, σε σχέση με τις υπόλοιπες διαστάσεις, προκειμένου να μην δημιουργείται σφάλμα λόγου μεγάλης διαφοράς στις τάξεις μεγέθους. Τελικά η υλοποίηση επετεύχθη ώστε η ακρίβεια των υπολογισμών, να προσεγγίζει πάντα το δεύτερο δεκαδικό ψηφίο για τις μονάδες των χιλιοστών (mm).

Τέλος να αναφερθεί ότι ο συνολικός κώδικας που διέπει την όλη εφαρμογή, ξεπέρασε τις 14000 σειρές. Ο προγραμματισμός του Inventor API αποτελεί το μεγαλύτερο μέρος των εντολών αυτών (περίπου 80% του συνόλου). Η αναφορά αυτή γίνεται με σκοπό να γίνει αντιληπτή η υπέρογκη εργασία που απαιτεί ένα Inventor API.

Δυσκολίες και περιορισμοί κατά τον Έλεγχο των Τιμών Εισόδου

Όσων αφορά τον έλεγχο των τιμών εισόδου δεν παρουσιάστηκαν ιδιαίτερες δυσκολίες. Απαίτηση ήταν η θεωρητική ανάλυση των μοντέλων των Κυλινδρικών Τομών. Αυτό είχε ως απαραίτητο την γνώση τριγωνομετρίας, την τρισδιάστατη γεωμετρική κατανόηση και κατ' επέκτασιν και την κατανόηση των δυσδιάστατων όψεων. Η ανάλυση αυτή τελικά ήταν υψίστης σημασίας αφού πέραν του ελέγχου των τιμών εισόδου, στην συνέχεια φάνηκε απαραίτητη και για την επίλυση προβλημάτων εκσφαλμάτωσης (Debugging). Έπειτα από την θεωρητική ανάλυση σε επίπεδο προγραμματισμού οι έλεγχοι πραγματοποιήθηκαν δίχως δυσκολίες με την χρήση της γλώσσας προγραμματισμού Visual Basic.

Η προαναφερθείσα θεωρητική ανάλυση λοιπόν, φάνηκε πολύ χρήσιμη κατά την διαδικασία προγραμματισμού και εκσφαλμάτωσης του Inventor API όπως έχει ήδη αναφερθεί. Πέραν αυτού, η εν λόγω θεωρητική ανάλυση αποδείχτηκε επιπλέον απαραίτητη και στις αποφάσεις που πάρθηκαν με σκοπό την βέλτιστη μοντελοποίηση και παρουσίαση του παραγόμενου μηχανολογικού σχεδίου. Σε προβλήματα γενικής μοντελοποίησης εντός ενός μηχανολογικού σχεδίου, δεν είναι πάντα δυνατή η αναλογική παρουσίαση. Για δυσανάλογες διαστάσεις ύψους και πλάτους πάντα δημιουργούνται προβλήματα παρουσίασης σε οριοθετημένο χώρο, γεγονός που συχνά δεν έχει λύση πέραν της ποιοτικής (μη αναλογικής) προβολής. Συνεπώς η θεωρητική ανάλυση των μοντέλων, επιπλέον, παρουσίασε ιδιαίτερη χρησιμότητα και κατά την λήψη αποφάσεων του πότε, πως και γιατί θα παρουσιάζονται ποιοτικά αναπτύγματα.

5.3 Πιθανές Επακόλουθες Μελλοντικές Εργασίες

Η συνολική εργασία υλοποιήθηκε να αφορά τον θεωρητικό υπολογισμό και την μοντελοποίηση των πέντε περιπτώσεων κυλινδρικών τομών όπως αυτές περιγράφηκαν. Όσον αφορά τις δυνατότητες ενός Inventor API είναι εφικτή η μοντελοποίηση και ο υπολογισμός των οποιοδήποτε τρισδιάστατων τομών ή γενικότερα εξαρτημάτων. Οι δυνατότητες και οι πιθανές υλοποιήσεις είναι απεριόριστες, ωστόσο η ανάπτυξη ενός οποιουδήποτε Inventor API δεν μπορεί να θεωρηθεί μελλοντική εργασία αυτής, αν δεν αποτελεί επακόλουθο ή βελτίωση της παρούσας. Στον παρόν υποκεφάλαιο λοιπόν θα περιγραφούν τρεις πιθανές επακόλουθες μελλοντικές εργασίες που είναι δυνατόν να παρέχουν επιπλέον δυνατότητες στο Inventor API που υλοποιήθηκε. Να αναφερθεί ότι όσον αφορά την μοντελοποίηση δεν υπάρχουν σημαντικά περιθώρια βελτίωσης μιας και αυτός ήταν

ένας από τους κύριους σκοπούς της εργασίας. Επομένως οι πιθανές επακόλουθες μελλοντικές εργασίες που θα περιγραφούν, αφορούν κυρίως περιθώρια βελτίωσης όσων αφορά την παροχή υπολογιστικών και μηχανολογικών δυνατοτήτων.

Υπολογισμός Με Μεταβλητή Σταθερά Ευλυγισίας

Όπως αναφέρθηκε η συνολική εργασία υλοποιήθηκε να αφορά τον θεωρητικό υπολογισμό των πέντε περιπτώσεων κυλινδρικών τομών. Ο υπολογισμός αυτός είναι θεωρητικός εφόσον δεν λήφθηκε υπόψιν η σταθερά ευλυγισίας. Το Autodesk Inventor εντός του υπο-περιβάλλοντος Sheet Metal Part χειρίζεται τρισδιάστατα μοντέλα θεωρώντας ότι αυτά έχουν δημιουργηθεί έπειτα από την δίπλωση ελασμάτων σταθερού πάχους. Στον κλάδο της Μηχανικής έχουν μελετηθεί τα ελάσματα σύμφωνα με την ευλυγισία τους. Το Autodesk Inventor λοιπόν έχει την δυνατότητα να υπολογίζει τα αναπτύγματα των ελασμάτων σύμφωνα με τις διαστάσεις που αυτά θα παρουσιάζουν στην πραγματικότητα, δεδομένου του πάχους τους και της σταθεράς ευλυγισίας του υλικού που τα αποτελεί. Επομένως μία μελλοντική εργασία θα μπορούσε να πραγματοποιεί τους πρακτικούς υπολογισμούς, προσφέροντας στον χρήστη την δυνατότητα της επιλογής της σταθεράς ευλυγισίας διάφορων τυποποιημένων υλικών του εμπορίου.

Μαζικός Υπολογισμός Κυλινδρικών Εξαρτημάτων

Οι Κυλινδρικές Τομές που υπολογίζονται στην εργασία που υλοποιήθηκε έχει αναφερθεί ότι παρουσιάζουν μηχανολογικό ενδιαφέρον σε σωληνώδη δίκτια μεταφοράς υγρών ή αερίων. Επομένως επακόλουθο αυτής της διπλωματικής εργασίας θα μπορούσε να είναι η τροποποίηση της με σκοπό το API να δέχεται μαζικά σετ πληροφοριών που χαρακτηρίζουν διαφορετικές διαστάσεις Κυλινδρικών Τομών, με σκοπό τον μαζικό υπολογισμό διαφορετικών εξαρτημάτων. Σημαντική δυνατότητα θα ήταν λοιπόν το κάθε σετ πληροφοριών που καθορίζει ένα εξάρτημα να μπορεί να ζητηθεί για συγκεκριμένη ποσότητα πολλών τέτοιον ίδιων, με σκοπό τον υπολογισμό του συνολικού απαιτούμενου εμβαδού, είτε του ωφέλιμου, είτε αυτού που τελικά θα πεταχτεί έπειτα από τις τομές όλων. Η προσθήκη αυτών των δυνατοτήτων θα ήταν χρήσιμες για τον υπολογισμό του συνολικού κόστους των ελασμάτων που ένα μαζικό έργο ενός σωληνοειδούς δικτιού απαιτεί.

Είσοδος Σε Μηχανήματα Δισδιάστατης Κοπής

Η βελτίωση της παρούσας εργασίας πέραν της παροχής χρήσιμων υπολογισμών, θα μπορούσε να αφορά και την πρακτική παροχή δυνατοτήτων στην μαζική παραγωγή εξαρτημάτων. Ένα δυσδιάστατο ανάπτυγμα όσων αφορά τον τομέα της πρακτικής Μηχανικής δεν είναι άλλο από ένα κομμένο έλασμα. Τα ελάσματα συνήθως απαραίτητο είναι να κόβονται από μηχανήματα δυσδιάστατης κοπής. Τα μηχανήματα αυτά λοιπόν δέχονται ένα σύνολο δεδομένων που αντιστοιχούν στον υπολογισμό των τιμών των αναπτυγμάτων. Ωστόσο δέχονται τα δεδομένα εισόδου τους με συγκεκριμένες μορφές αρχείων. Επομένως επακόλουθο αυτής της διπλωματικής εργασίας θα μπορούσε να είναι η τροποποίηση της με σκοπό να φέρει επιπλέον την δυνατότητα παραγωγής αρχείων συμβατών με μηχανήματα δυσδιάστατης κοπής. Τα μηχανήματα δυσδιάστατης κοπής απαιτούν μεγάλο πλήθος υπολογισμών το οποίο είναι ανάλογο με το συνολικό μήκος των τομών που πρόκειται να υλοποιήσουν. Συνεπώς το πλήθος των υπολογισμών αυτών θα ήταν τελικά χρήσιμο να αποφασίζεται ανάλογα με της διαστάσεις του εκάστοτε υπολογισμού και σε συνδυασμό με την παροχή της επιλογής της πυκνότητας σημείων (πλήθος υπολογισμένων σημείων ανά μονάδα απόστασης).

Συνδυασμός Των Προαναφερθέντων Σε Ενιαία Εφαρμογή

Φυσικά και δεν θα μπορούσε να μην αναφερθεί η δυνατότητα βελτίωσης της εργασίας σε επίπεδο συνδυασμού όλων των επιλογών βελτίωσης που έχουν περιγραφεί. Συνεπώς ιδανική θα ήταν η τροποποίηση της εφαρμογής που δημιουργήθηκε, προκειμένου να επιλύει βιομηχανικά πρακτικούς υπολογισμούς, μαζικά για πολλά εξαρτήματα και σύμφωνα με τον συντελεστή ευλυγισίας διάφορων τυποποιημένων υλικών του εμπορίου. Η εν λόγω τροποποίηση θα ήταν επιπλέον χρήσιμο να υπολογίζει συνολικές επιφανείς (ωφέλιμες και μη) καθώς και συνολικά χρηματικά κόστη. Τέλος, βέλτιστο θα ήταν η ίδια εφαρμογή να είναι παράλληλα συμβατή με μηχανήματα δυσδιάστατης κοπής και να έχει την δυνατότητα να δρομολογεί ενέργειες σε αυτά.

Πηγές

[1] Βιβλίο "Μηχανολογικό σχέδιο" - Αριστομένης Θ. Αντωνιάδης - 3η Έκδοση | Κεφάλαιο 12. URL:http://www.antoniadis.gr/index.html

[2] Autodesk University | Creating Add-Ins for Inventor
URL:https://www.autodesk.com/autodesk-university/class/Creating-Add-Ins-Inventor-2018

[3] Autodesk University 2008 | How Deep is the Rabbit Hole? Examing the Matrix and other Inventor® Math and Geometry Objects.
URL:https://forums.autodesk.com/autodesk/attachments/autodesk/22/9266/1/Math Geometry.pdf

[4] Autodesk Forums | Inventor iLogic and VB.net Forum
URL:https://forums.autodesk.com/t5/inventor-ilogic-and-vb-net-forum/bd-p/120

[5] Autodesk Help | Autodesk Inventor 2018 API Help
URL:https://help.autodesk.com/view/INVNTOR/2018/ENU/?guid=GUID-6FD7AA08-1E43-43FC-971B-5F20E56C8846

[6] Microsoft Learn | Windows Forms Documentation
URL:https://learn.microsoft.com/en-us/dotnet/desktop/winforms

[7] Microsoft Learn | Visual Studio IDE documentation
URL:https://learn.microsoft.com/en-us/visualstudio/ide/?view=vs-2022

[8] Microsoft Learn | Windows Forms Application Basics (Visual Basic)
URL:https://learn.microsoft.com/en-us/dotnet/visual-basic/developing-apps/windows-forms/

[9] Microsoft Learn | Create a WinForms app with Visual Basic
URL:https://learn.microsoft.com/en-us/visualstudio/ide/create-a-visual-basic-winform-in-visual-studio?view=vs-2022