Ιόνιο Πανεπιστήμιο

Τμήμα Πληροφορικής

-- Πτυχιακή Εργασία --

Διεπαφή χρήστη διαχείρισης και εκτέλεσης εφαρμογών μέσω πολυτροπικών περιφερειακών συσκευών

Κωνσταντίνος Τουρτσάκης, Π2019140

Επιβλέπων: – Μιχαήλ Στεφανιδάκης –

June 8, 2024

Επιβλέπων

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο Τμήμα Πληροφορικής

Τριμελής Επιτροπή

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο Τμήμα Πληροφορικής
Δημήτριος Ρίγγας, ΕΔΙΠ,
Ιόνιο Πανεπιστήμιο Τμήμα Πληροφορικής
Θεόδωρος Ανδρόνικος, Αναπληρωτής Καθηγητής,
Ιόνιο Πανεπιστήμιο Τμήμα Πληροφορικής

Περίληψη

To do...

Contents

List of Figures

List of Tables

Chapter A

Εισαγωγή

Είναι πλέον σίγουρο πως η χρήση ενός προσωπικού υπολογιστή γίνεται μέσω του πληκτρολογίου σε συνδυασμό με την χρήση του ποντικιού για την περιήγηση του χρήστη μέσα στο γραφικό περιβάλλον των σύγχρονων συστημάτων. Όμως αυτό ήταν ανέκαθεν μια μέθοδος περιήγησης που προορίζονταν για εργασιακή χρήση. Με την εξέλιξη των τεχνολογιών και την εισαγωγή ολοένα και περισσότερων πολυμεσικών εφαρμογών στο περιβάλλον του Η/Υ, ήταν αναπόφευκτη η μετάβαση σε μια εποχή όπου ο Η/Υ έχει εφαρμογή σε κάθε σπίτι ανεξαρτήτως του επαγγέλματος του ιδιωκτήτη. Αναγνωρίζοντας την αλλαγή αυτή και το γεγονός πως παραμένουν περιθώρια βελτίωσης από την πλευρά του συστήματος ώς προς την επικοινωνία μεταξύ του χρήστη και του υπολογιστή, το πρόγραμμα αυτό έχει ως στόχο την εκτέλεση εφαρμογών ανεξαρτήτος της συσκευής εισόδου του χρήστη. Επομένως, απώτερος σκοπός είναι η αξιοποίηση των νέων περιφειακών συσκευών στην εκτέλεση και περιήγηση του λειτουργικού συστήματος αλλά και την υλοποίηση νέων διαδικασιών εκτέλεσης εφαρμογών από προϋπάρχουσες συσκευές, όπως το πληκτρολόγιο και το ποντίκι. Για την υλοποίηση αυτής της εφαρμογής είναι απαραίτητο να γίνει χρήση τεχνολογιών για την γραφική διεπαφή χρήστη και την είσοδο δεδομένων από τις περιφερειακές συσκευές στο σύστημα. Για την εξυπηρέτηση αυτών των αναγκών γίνεται κυρίως χρήση της βιβλιοθήκης Qt6, του XInput και της βιβλιοθήκης των Windows.

Chapter B

Ανάπτυξη εφαρμογών σε Qt6

B.1 Τι είναι το Qt;

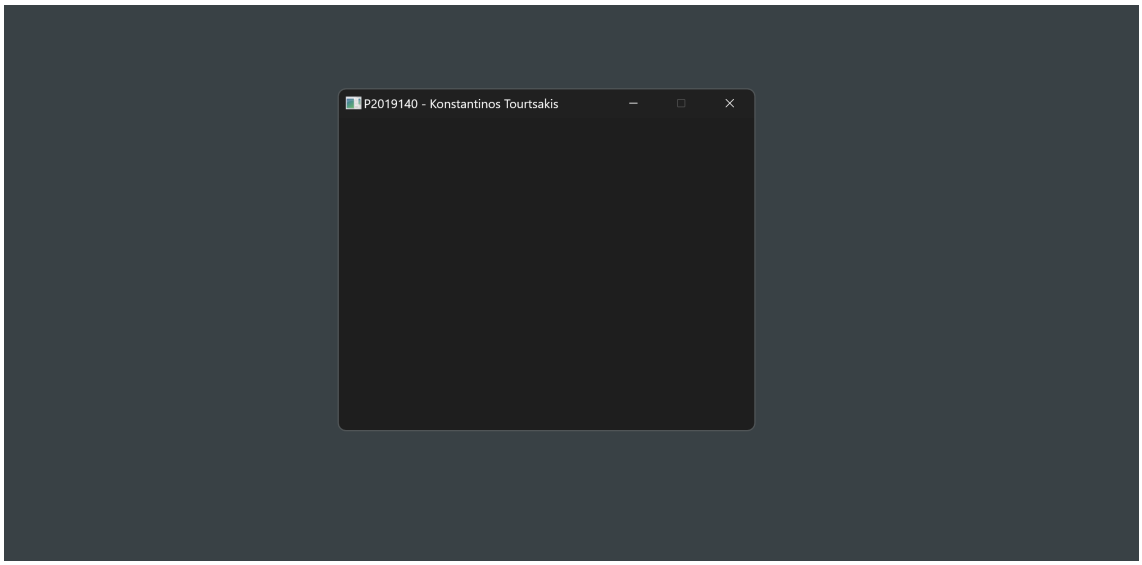
Το Qt είναι μια δημοφιλής βιβλιοθήκη ανάπτυξης εφαρμογών διεπαφής χρήστη το οποίο είναι διαθέσιμο σε C++ και Python. Με το Qt είναι εφικτή η υλοποίηση cross-platform εφαρμογών με το τελικό αποτέλεσμα να είναι μια αξιοπρεπής διεπαφή χρήστη που λειτουργεί αποτελεσματικά και αξιόπιστα, υποστηρίζοντας μια πληθώρα συστημάτων όπως Windows, Linux, MacOS, Android και ενσωματωμένα συστήματα. Παρακάτω γίνεται περιγραφή της λειτουργίας της βιβλιοθήκης αυτής όπου στην περίπτωση αυτή θα γίνει χρήση παραδειγμάτων σε γλώσσα προγραμματισμού C++ μιας και είναι η γλώσσα στην οποία έχει γραφεί το παρόν πρόγραμμα.

B.2 Ένα απλό πρόγραμμα σε Qt6

Για την δημιουργία ενός παραθύρου Qt6 πρέπει πρώτα να δημιουργηθεί ένα αντικείμενο τύπου QApplication και στην συνέχεια να αρχικοποιηθεί ένα αντικείμενο QWidget το οποίο θα αποτελεί το παράθυρο της εφαρμογής πάνω στο οποίο θα προστεθούν τα υπόλοιπα στοιχεία της γραφικής διεπαφής για τις λειτουργίες του προγράμματος. Παρακάτω βλέπουμε

ένα παράδειγμα με την αντίστοιχη περιγραφή μεταφρασμένη σε κώδικα.

```
#include <QApplication>
#include <QWidget>


class MyWidget : public QWidget
{
public:
    MyWidget(QWidget *parent = nullptr) : QWidget(parent)
    {
        setFixedSize(400, 300);
        setWindowTitle("P2019140 - Konstantinos Tourtsakis");
    }
};

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    MyWidget widget;
    widget.show();

    return app.exec();
}
```



Η κλάση MyWidget κληρονομεί την κλάση QWidget. Το παράθυρο το οποίο δημιουργείται εμφανίζεται σε πλήρες μέγεθος και στην συνέχεια επιστρέφεται το αντικείμενο της εφαρμογής το οποίο το διαχειρίζεται η βιβλιοθήκη κατά την έξοδο εκτέλεσης του προγράμματος.

B.3 Προσθήκη στοιχείων γραφικής διεπαφής σε Qt6

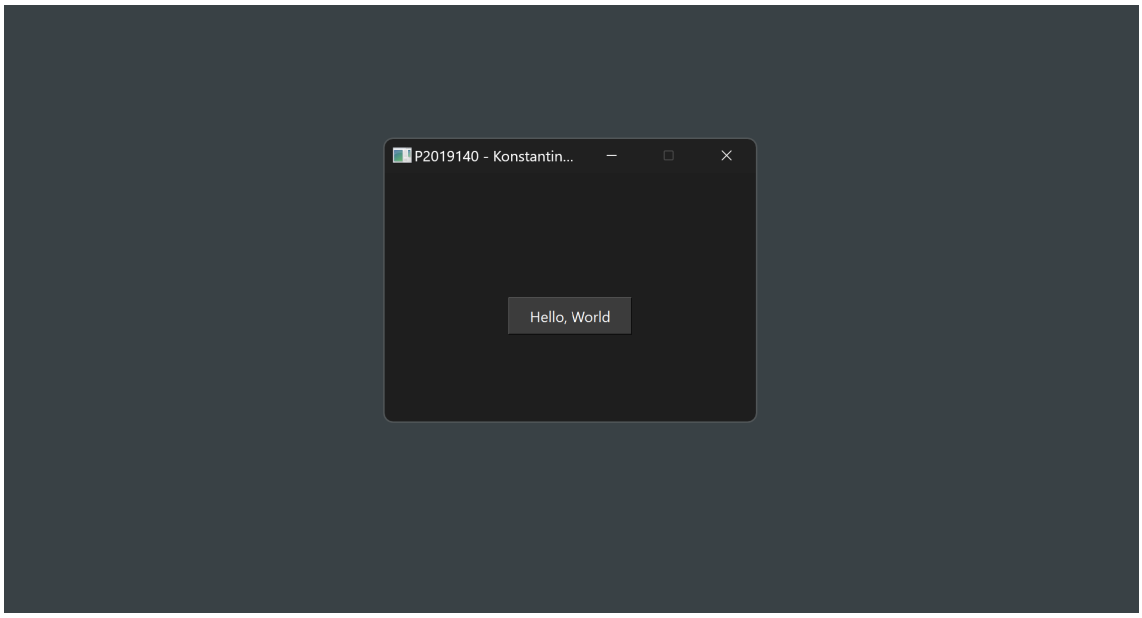
Κάθε εφαρμογή γραφικής διεπαφής παρέχει στοιχεία μέσω των οποίων γίνεται η διαχείριση των δεδομένων που επεξεργάζεται και η εκτέλεση των λειτουργιών του. Συνήθως τα δεδομένα αυτά δεν είναι τίποτε άλλο από τους βασικούς τύπους δεδομένων που υποστηρίζουν οι γλώσσες προγραμματισμού: int, bool, float και string. Επιπλέον υπάρχουν στοιχεία με τα οποία εξυπηρετήται αποτελεσματικότερα ο σκοπός του προγράμματος, είτε λόγω ευκολίας είτε λόγω κατανόησης από τον χρήστη. Παρακάτω βλέπουμε τα στοιχεία που αξιποιεί το πρόγραμμα της εργασίας για την επίτευξη του σκοπού του.

B.3.1 Δημιουργία QPushButton

Ένα QPushButton είναι ένα κουμπί το οποίο έχει την ιδιότητα εκτέλεσης λειτουργιών. Για την προσθήκη μιας λειτουργίας πάνω στο κουμπί αυτό χρειάζεται να γίνει η σύνδεση

μεταξύ του αντικειμένου αυτού και μιας μεθόδου που ανήκει στην κλάση που κληρονομεί το QWidget. Ορίζεται το signal το οποίο υποστηρίζει το κάθε αντικείμενο μέσω του οποίου θα γίνει η κλήση της μεθόδου που ονομάζεται slot από το Qt. Τα signals που υποστηρίζουν τα στοιχεία του Qt είναι διαφορετικά, ανάλογα με τους στόχους που προσπαθεί να πετύχει το κάθε ένα από αυτά. Επομένως ένα στοιχείο QPushButton μπορεί να έχει περισσότερα ή λιγότερα singals σε σύγκριση με ένα QComboBox. Παρακάτω βλέπουμε ένα παράδειγμα σε κώδικα.

```
class MyWidget : public QWidget
{
public:
    MyWidget(QWidget *parent = nullptr) : QWidget(parent)
    {
        setFixedSize(300, 200);
        setWindowTitle("P2019140 - Konstantinos Tourtsakis");

        QPushButton *button = new QPushButton("Hello, World", this);
        button->setGeometry(100, 100, 100, 30);

        connect(button, &QPushButton::clicked, this, &MyWidget::PrintHello);
    }

public slots:
    void PrintHello()
    {
        std::cout << "Hello, World!" << std::endl;
    }
};
```



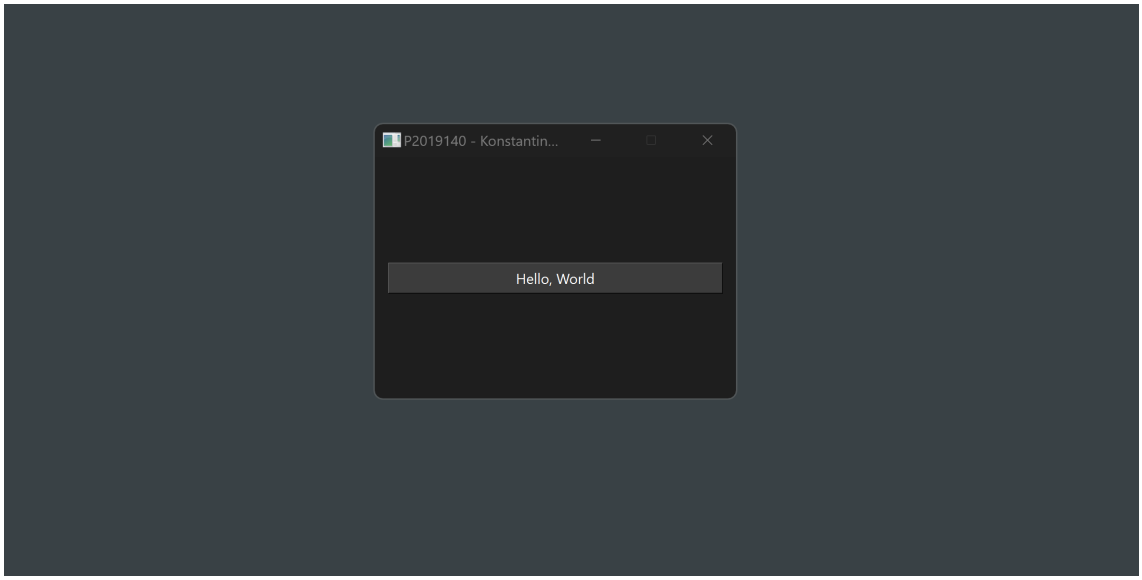
Στο παράδειγμα γίνεται ή δημιουργία και ή αρχικοποίηση κουμπιού με το όνομά του και στην συνέχεια η σύνδεση. Στην σύνδεση καλείται η μέθοδος connect στην οποία ορίζεται το signal το οποίο θα πυροδοτήσει την κλήση του slot που έχει ανατεθεί στο αντικείμενο. Στην προκειμένη περίπτωση έχουμε ορίσει το QPushButton::clicked signal το οποίο συνδέει το κουμπί με την μέθοδο PrintHello.

B.3.2 Δημιουργία γραφικού πλαισίου

Ένα layout είναι ένα πλαίσιο στο οποίο μπορούν να τοποθετηθούν άλλα στοιχεία του Qt, όπως το QPushButton που προαναφέρθηκε. Το Qt6 παρέχει 3 βασικά είδη layouts. Το QVBoxLayout, το QHBoxLayout και το QGridLayout. Τα πρώτα δύο παρέχουν ένα παρόμοιο πλαίσιο με την μόνη τους διαφορά να βρίσκεται στην κατεύθυνση των στοιχείων μέσα στο πλαίσιο. Επομένως, ένα QVBoxLayout χρησιμοποιείται για στοιχεία που θα τοποθετηθούν κάθετα (vertical) και ένα QHBoxLayout θα χρησιμοποιηθεί για στοιχεία που πρόκειται να τοποθετηθούν όριζόντια (horizontal). Τέλος, ένα QGridLayout χρησιμοποιείται για την τοποθέτηση στοιχείων σε μορφή πίνακα. Το πλαίσιο αυτό διαθέτει θέσεις που αναπαριστούν ένα σημείο σε έναν δισδιάστατο χώρο. Κάθε σημείο έχει μια θέση η οποία

είναι μοναδική και είναι προσβάσιμη μέσω της τιμής της σειράς και της στήλης στην οποία βρίσκεται. Παρακάτω βλέπουμε κώδικα με την χρήση ενός QVBoxLayout και την προσθήκη ενός QPushButton σε αυτό.

```
    MyWidget(QWidget *parent = nullptr) : QWidget(parent)
{
    setFixedSize(300, 200);
    setWindowTitle("P2019140 - Konstantinos Tourtsakis");

    QVBoxLayout *layout = new QVBoxLayout(this);

    QPushButton *button = new QPushButton("Hello, World", this);
    button->setGeometry(100, 100, 100, 30);

    layout->addWidget(button);

    connect(button, &QPushButton::clicked, this, &MyWidget::PrintHello);
}
```

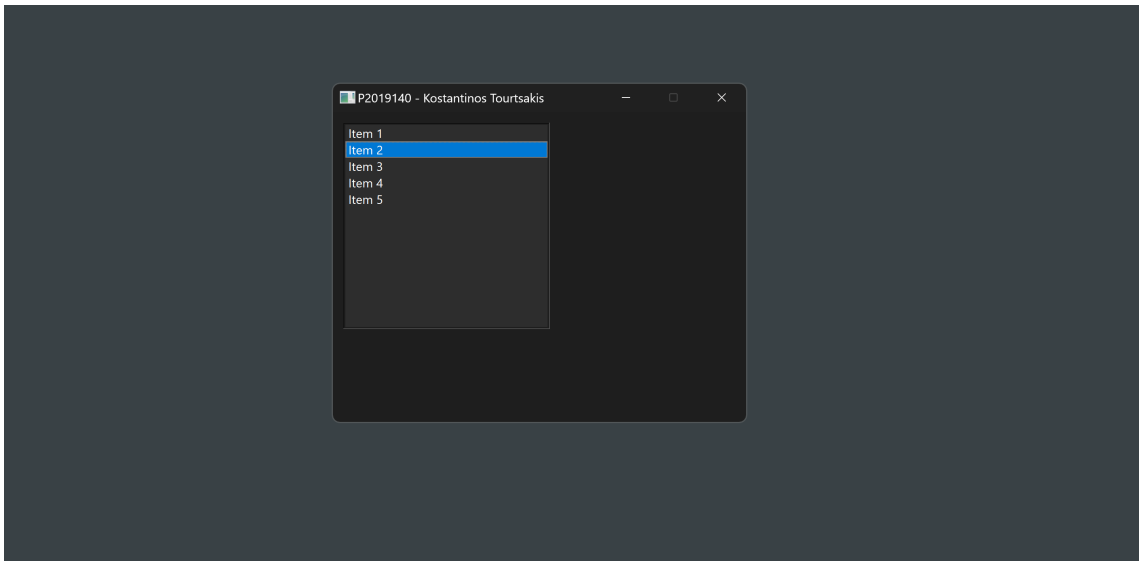


Κάθε στοιχείο τύπου QWidget προστίθεται πάνω στο layout με την κλήση της μεθόδου addWidget και αντίστοιχα η αφαίρεση του γίνεται με την μέθοδο removeWidget.

B.3.3 Δημιουργία λίστας αντικειμένων

Μια λίστα QWidget μπορεί να αποθηκεύσει στην μνήμη στοιχεία τύπου QListWidgetItem. Τα στοιχεία αυτά είναι αντικείμενα του Qt τα οποία δεν μπορούν να αποθηκευθούν σε κάποια άλλη

```
#include <QListWidget>
#include <QListWidgetItem>


class MyWidget : public QWidget
{
public:
    MyWidget(QWidget *parent = nullptr) : QWidget(parent)
    {
        setFixedSize(400, 300);
        setWindowTitle("P2019140 - Konstantinos Tourtsakis");

        QListWidget *list_widget = new QListWidget(this);
        list_widget->addItem(new QListWidgetItem("Item 1"));
        list_widget->addItem(new QListWidgetItem("Item 2"));
        list_widget->addItem(new QListWidgetItem("Item 3"));
        list_widget->addItem(new QListWidgetItem("Item 4"));
        list_widget->addItem(new QListWidgetItem("Item 5"));
        list_widget->setGeometry(10, 10, 200, 200);
    }
};
```

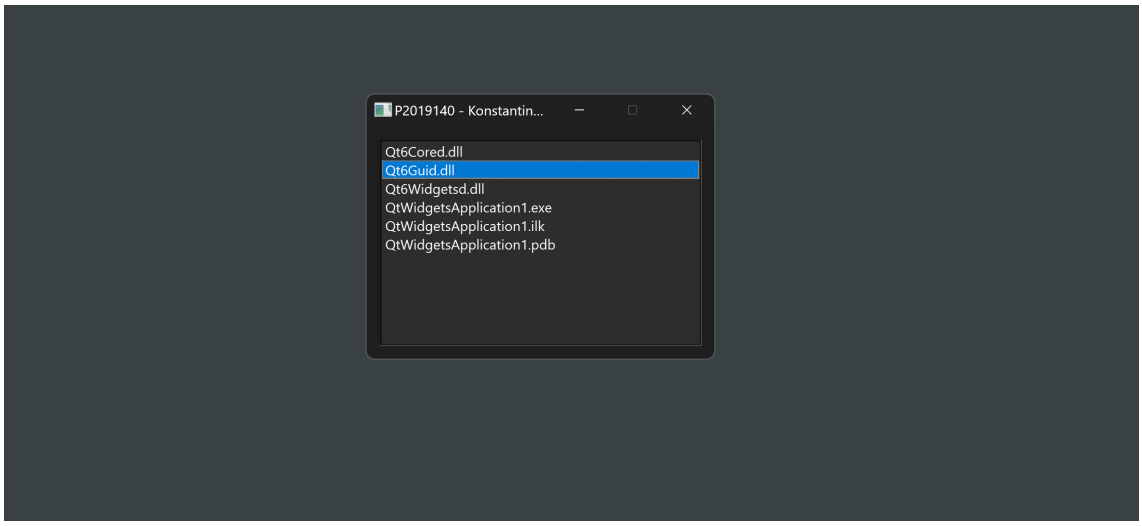


B.3.4 Διάβασμα αρχείων από directory

Το διάβασμα αρχείων από ένα directory γίνεται μέσω της QDir κλάσης στην οποία αρχικοποιείται ένα αντικείμενο με το path του directory του οποίου θέλουμε να διαβάσουμε. Στην συνέχεια αποθηκεύουμε τα αρχεία του directory σε μια λίστα από QStrings και τα προσπελαύνουμε για την προσθήκη τους σε ένα QListWidget με στόχο την προβολή τους στον χρήστη.

```
#include <QDir>
#include <QStringList>


class MyWidget : public QWidget
{
public:
    MyWidget(QWidget* parent = nullptr) : QWidget(parent)
    {
        setFixedSize(300, 200);
        setWindowTitle("P2019140 - Konstantinos Tourtsakis");

        QVBoxLayout* layout = new QVBoxLayout(this);
        QListWidget* list_widget = new QListWidget(this);
```

```
        layout->addWidget(listWidget);

        QDir directory("C:\\Users\\kosta\\Documents\\Git\\Thesis\\source\\x64\\Debug");

        QStringList files = directory.entryList(QDir::Files);
        for (const QString& file : files)
        {
            list_widget->addItem(file);
        }
    }
};
```

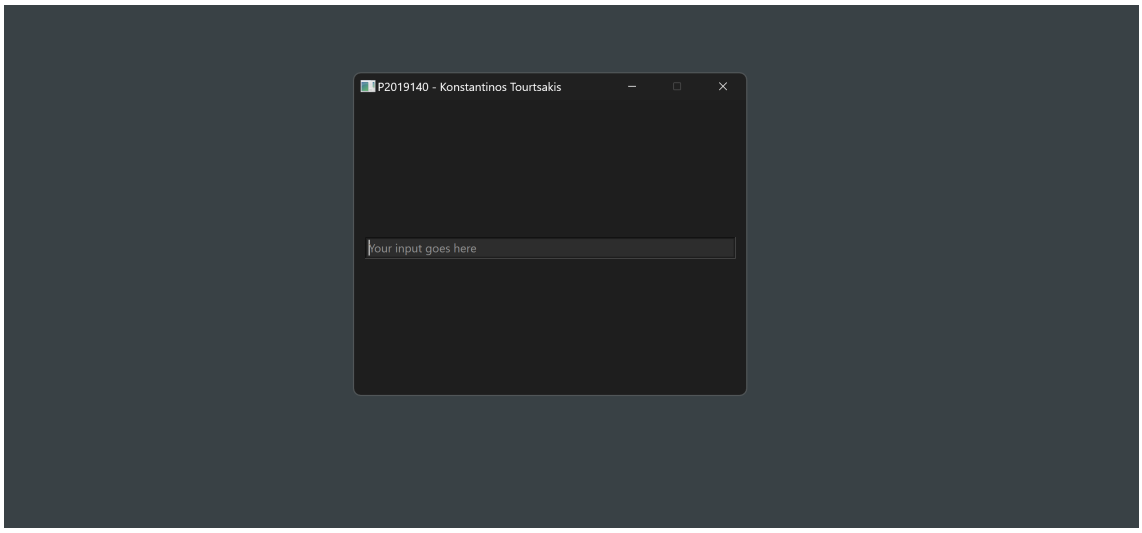


B.3.5 Δημιουργία input field

Βασικό στοιχείο κάθε εφαρμογής γραφικής διεπαφής αποτελεί ένα input field μιας και επιτρέπει στον χρήστη να πληκτρολογήσει δεδομένα εισόδου για την εκτέλεση μιας λειτουργίας του προγράμματος. Για τον σκοπό αυτό το Qt παρέχει τα αντικείμενα τύπου QLineEdit. Όπως και τα QPushButton, τα αντικείμενα αυτά αρχικοποιούνται και στην συνέχεια προστίθενται πάνω σε ένα πλαίσιο μέσω της μεθόδου addWidget.

```
#include <QLineEdit>
```

```
class MyWidget : public QWidget
{
public:
    MyWidget(QWidget* parent = nullptr) : QWidget(parent)
    {
        setFixedSize(400, 300);
        setWindowTitle("P2019140 - Konstantinos Tourtsakis");

        QLineEdit* lineEdit = new QLineEdit(this);
        lineEdit->setPlaceholderText("Your input goes here");

        QVBoxLayout* layout = new QVBoxLayout(this);
        layout->addWidget(lineEdit);

        setLayout(layout);
    }
};
```



B.4 Αποθήκευση στοιχείων στον δίσκο

Κάθε εφαρμογή στις ημέρες μας χρειάζεται να αποθηκεύει την κατάσταση στην οποία βρισκόταν την τελευταία φορά που εκτελέστηκε με στόχο να γίνει πιο κατανοητή η χρήση του. Για τον σκοπό αυτό, το Qt φροντίζει να παρέχει τα απαραίτητα εργαλεία για την αποθήκευση των στοιχείων της εφαρμογής βελτιώνοντας με αυτόν τον τρόπο τον πηγαίο κώδικα που γράφεται για την προσθήκη της λειτουργίας αυτής. Παρακάτω βλέπουμε μια μέθοδο που αποθηκεύει τα στοιχεία ενός Qt6 προγράμματος.

```
void SaveSettings()
{
    QSettings settings("Company_name", "Application_title");
    settings.beginGroup("group_name");

    settings.setValue("WindowSize", window()->size());
    settings.setValue("Username", "JohnDoe");

    settings.endGroup("group_name");
}
```

Αρχικά δημιουργείται ένα αντικείμενο τύπου QSettings. Ακολουθεί ο ορισμός της ομάδας στην οποία θα γίνουν οι αλλαγές και στην συνέχεια ορίζονται οι τιμές πάνω στις οποίες θα γραφούν οι μεταβλητές του προγράμματος. Στο τέλος επαναρχικοποιείται η ομάδα καλόντας την μέθοδο endGroup.

B.5 Ανάκτηση εικονιδίων συστήματος αρχείων

Όπως είναι γνωστό τα γραφικά περιβάλλοντα στηρίζονται πολύ στην χρήση εικονιδίων για την διευκόλυνση του χρήστη κατά την περιήγησή του μιας και είναι πολύ πιο αποτελεσματική η συσχέτιση μιας λειτουργίας με μια εικόνα παρά με κάποιο κείμενο. Επομένως, είναι σημαντικό να υπάρχει μια συνάρτηση που επιστρέφει το εικονίδιο ενός αρχείου δυναμικά, ώστε ο χρήστης να καταλαβαίνει από την αρχή την λειτουργία που κρύβεται πίσω από

την ενεργοποίηση του εικονιδίου αυτού. Παρακάτω βλέπουμε την συνάρτηση αυτή.

```
QIcon GetFileIcon(const QString& file_path)
{
    SHFILEINFO shfi;
    memset(&shfi, 0, sizeof(SHFILEINFO));

    if (SHGetFileInfo(reinterpret_cast<const wchar_t*>(file_path.utf16()), 0, &shfi, sizeof(
    {
        QPixmap pixmap = QPixmap::fromImage(QImage::fromHICON(shfi.hIcon)).scaled(QSize(Perc
        QIcon icon(pixmap);

        // Cleanup the icon resource
        DestroyIcon(shfi.hIcon);

        return icon;
    }


    return QIcon();
}
```

Επειδή η εφαρμογή στην οποία αναφέρεται το παρόν κείμενο προορίζεται για χρήση σε συστήματα Windows, η συγκεκριμένη συνάρτηση στηρίζεται πάνω σε αυτά. Αρχικά δημιουργείται ένα struct τύπου SHFILEINFO πάνω στο οποίο θα αποθηκευτούν τα δεδομένα του αρχείου, συμπεριλαμβανομένου και του εικονιδίου. Στην συνέχεια γίνεται έλεγχος για την ανάκτηση του εικονιδίου από το σύστημα. Αφού η ανάκτηση γίνει με επιτυχία, δημιουργείται ένα QPixmap με το εικονίδιο του αρχειού. Στην συνέχεια το pixmap χρησιμοποιείται για την δημιουργία του QIcon εικονιδίου το οποίο και θα επιστραφεί από την συνάρτηση. Αν το εικονίδιο δεν βρεθεί η συνάρτηση θα επιστρέψει ένα προεπιλεγμένο QIcon.

B.6 Προτάσεις δεδομένων εισόδου σε input field

Συχνά οι χρήστες βρίσκουν χρήσιμη την υποβοήθησή τους από την γραφική διεπαφή στην περιήγησή τους στο εκάστοτε πρόγραμμα. Μια τέτοια βοήθεια μπορεί να συμβεί επίσης στην είσοδο δεδομένων σε ένα input field. Ανάλογα με τις ενέργειες του χρήστη υπάρχει κάποιος αλγόριθμος που καθορίζει την λίστα από την οποία το input field θα προτείνει στον χρήστη δεδομένα εισόδου. Στο Qt αυτή η λειτουργία υλοποιείται με ένα αντικείμενο QCompleter που ουσιαστικά συμπληρώνει αυτό που έχει γράψει ο χρήστης με προτάσεις δεδομένων.

```
QLineEdit input_field;
QStringList suggestions;
suggestions << "Suggestions" << "for" << "thesis";

QCompleter* completer = new QCompleter(suggestions, &input_field);
completer->setCaseSensitivity(Qt::CaseInsensitive);

input_field.setCompleter(completer);
```

B.7 Επαναληπτική εκτέλεση διεργασιών παρασκηνίου

Ένα βασικό πρόβλημα με τις βιβλιοθήκες εφαρμογών γραφικής διεπαφής είναι το γεγονός πως αμέσως μετά την δημιουργία των γραφικών στοιχείων η βιβλιοθήκη αποκτά τον πλήρη έλεγχο της εφαρμογής με αποτέλεσμα να μην είναι εφικτή η εκτέλεση κώδικα επαναληπτικά. Αυτό καθιστά αδύνατη την εκτέλεση διεργασιών ρουτίνας οι οποίες πρέπει να εκτελούνται σε κάθε επανάληψη του προγράμματος ώστε να γίνει εφικτή η υλοποίηση ορισμένων λειτουργιών. Ευτυχώς οι περισσότερες βιβλιοθήκες, όπως και το Qt, διαθέτουν αντικείμενα που επιτρέπουν την προσθήκη κώδικα που εκτελείται σε τακτά χρονικά διαστήματα που ορίζονται από τον προγραμματιστή. Αυτό όπως είναι κατανοητό μπορεί να αξιοποιηθεί ορίζοντας ένα μηδενικό χρονικό διάστημα, δημιουργόντας κατά κάποιο τρόπο μια επανάληψη που την διαχειρίζεται το Qt.

```
QTimer* timer = new QTimer(this);
connect(timer, &QTimer::timeout, this, &MyWidget::TaskMainUserLoop);
timer->start(0);
```

Chapter C

Η βιβλιοθήκη XInput

C.1 Τι είναι το XInput;

Το XInput είναι μια βιβλιοθήκη διαχείρισης εντολών εισόδου για συσκευές χειρισμού βιντεοπαιχνιδιών κατασκευασμένες από την Microsoft. Οι συγκεκριμένες συσκευές είναι χειριστήρια Xbox οποιασδήποτε γεννιάς. Με το XInput έχουμε την δυνατότητα να διαχειριστούμε εντολές εισόδου από τα κουμπιά του χειριστηρίου και κατ'επέκταση να προσθέσουμε την δική μας λειτουργικότητα κατά την πίεση των κουμπιών αυτών. Στην συνέχεια βλέπουμε την κλάση στην οποία ορίζονται οι μέθοδοι που αξιοποιούνται για τον σκοπό αυτό.

C.2 Η κλάση του χειριστηρίου

Για την διαχείρηση των εντολών εισόδου του χειριστηρίου έχει δημιουργηθεί η αντίστοιχη κλάση. Η κλάση περιλαμβάνει δεδομένα που αφορούν καταστάσεις του χειριστηρίου, μεθόδους σχετικά με τον έλεγχο καταστάσεων των κουμπιών του αλλά και μεθόδους που ελέγχουν την συνδεσιμότητα της συσκευής. Παρακάτω βλέπουμε αναλυτικότερα τον ορισμό της κλάσης και τα στοιχεία που την απαρτίζουν.

```
class XBOXController
{
```

```
private:
    XINPUT_STATE controller_state;
    int controller_number;
public:
    float left_trigger = 0.0f;
    float right_trigger = 0.0f;

    XBOXController(int playerNumber);
    XINPUT_STATE GetState();
    bool IsConnected();
    void Vibrate(const int left_val = 0, const int right_val = 0);
    SHORT GetLeftAnalogX();
    SHORT GetLeftAnalogY();
    SHORT GetRightAnalogX();
    SHORT GetRightAnalogY();
    bool IsButtonDown(const int button);
    bool IsButtonJustUp(const int button);
    bool IsButtonJustDown(const int button);
};
```

Chapter D

Μια λεπτομερής περιγραφή της εφαρμογής

D.1 Περιγραφή λογισμικού

Το λογισμικό αυτό αποτελείται από 3 μέρη τα οποία ουσιαστικά αποτελούν και τις περιφερειακές συσκευές τις οποίες υποστηρίζει για την επίλυση του προβλήματος. Αυτά είναι το ποντίκι, το πληκτρολόγιο και το χειριστήριο κονσόλας βιντεοπαιχνιδιών. Προτού όμως ξεκινήσει η αποκλειστική χρήση του συστήματος με μόνο μία από αυτές τις περιφερειακές συσκευές θα χρειαστεί να γίνει η αρχικοποίηση της εφαρμογής κατά την πρώτη εκτέλεση του προγράμματος από τον χρήστη. Το πρόγραμμα θα καθοδηγήσει τον χρήστη θέτοντας μέσω των γραφικών στοιχείων κάποιες απαραίτητες για την εκτέλεση τις εφαρμογής παραμέτρους. Οι παράμετροι αυτοί είναι το όνομα του προφίλ του χρήστη που θα το χρησιμοποιήσει και ένα αρχικό directory στο οποίο βρίσκονται οι περισσότερες εφαρμογές που είναι εγκατεστημένες στον υπολογιστή. Με αυτόν τον τρόπο η εφαρμογή θέτει τις προεπιλεγμένες ρυθμίσεις για το προφίλ του χρήστη και φτάνει στο τελικό στάδιο που είναι και βασικός στόχος της, δηλαδή η εκτέλεση εφαρμογών.

D.2 Εκτέλεση εφαρμογών με το ποντίκι

Το ποντίκι όπως είναι γνωστό είναι ένα εξαιρετικά απλό μέσο με το οποίο επικοινωνεί ο χρήστης με τον υπολογιστή και επιτρέπει την χρήση ενός μόνο χεριού για την περιήγηση μέσα σε αυτόν. Επομένως, η υποστήριξη της συσκευής αυτής από την εφαρμογή γίνεται μέσω της γραφικής διεπαφής από όπου και είναι εύκολα προσβάσιμα τα στοιχεία που διαχειρίζονται την εκτέλεση, προσαρμογή και διαχείρηση των προγραμμάτων που είναι εγκατεστημένα στο περιβάλλον του χρήστη. Για τον σκοπό αυτό έχουν προστεθεί τα κατάλληλα στοιχεία γραφικής διεπαφής που υλοποιούν την προβλεπόμενη από τον τελικό χρήστη λειτουργικότητα της εφαρμογής. Πιο συγκεκριμένα, η γραφική διεπαφή έχει χωριστεί σε τρία κομμάτια. Αυτά υλοποιούνται μέσω του Qt σε καρτέλες οι οποίες στην συνέχεια προστίθενται στο κεντρικό QTabWidget που αποτελεί κομμάτι του βασικού γραφικού στρώματος του προγράμματος. Για τις ανάγκες της εφαρμογής έχει δημιουργηθεί μια καρτέλα με τις εφαρμογές που γνωρίζει η εφαρμογή, μια καρτέλα με τις αγαπημένες εφαρμογές του χρήστη και μια καρτέλα με τις ρυθμίσεις που μπορεί να αλλάξει ο χρήστης για να φέρει την λειτουργικότητα του προγράμματος πιο κοντά με τις προσωπικές του προτιμήσεις.

```
tabs->addTab(tab_all_apps, "Applications");
tabs->addTab(tab_favorites, "Favorites");
tabs->addTab(tab_settings, "Settings");

layout_root->addWidget(tabs);
```

Ας αναλύσουμε τώρα την κάθε καρτέλα ξεχωριστά ξεκινώντας από την καρτέλα των ρυθμίσεων για λόγους συνοχής.

D.2.1 Καρτέλα ρυθμίσεων

Στην καρτέλα με τις ρυθμίσεις βρίσκονται όλες οι λειτουργίες οι οποίες προσαρμόζουν το πρόγραμμα στις ανάγκες του χρήστη και λειτουργίες που του επιτρέπουν να διαχειριστεί το προφίλ του. Επιπλέον, μέσα από αυτήν την καρτέλα ο χρήστης αποκτά πρόσβαση στον εικονικό πληκτρολόγιο μέσω ενός κουμπιού το οποίο αφορά την περιήγηση μέσω

ποντικιού (το εικονικό πληκτρολόγιο είναι επίσης προσβάσιμο από το χειριστήριο κονσολών βιντεοπαιχνιδιών αλλά η πρόσβαση σε αυτό γίνεται με διαφορετικό τρόπο). Στην συνέχεια υπάρχουν ρυθμίσεις που αφορούν τις λίστες με τις εφαρμογές του χρήστη που εμφανίζονται στις καρτέλες Applications και Favorites. Υπάρχει το checkbox ”Include File Extension” το οποίο συμπεριλαμβάνει την κατάληξη του αρχείου της εφαρμογής στο όνομα που εμφανίζεται στην λίστα. Στην συνέχεια υπάρχουν checkboxes που επηρρεάζουν τον τρόπο εμφάνισης των λιστών. Αυτά είναι τα ”Icon Mode” και ”Wrapping” που καθορίζουν αν η λίστες θα εμφανίζονται σε εικονίδια ή όχι και αντίστοιχα αν τα στοιχεία της κάθε λίστας εμφανίζονται δίπλα στα υπόλοιπα όταν δεν υπάρχει πλέον χώρος στην ορατή περιοχή. Στην συνέχεια υπάρχει ένα combo box μέσα από το οποίο ο χρήστης μπορεί να επιλέξει την θεματολογία που επιθυμεί ανάλογα με τις προτιμήσεις χρωμάτων που έχει. Η εφαρμογή παρέχει τις θεματολογίες ”dark”, ”light”, ”red”, ”green”, ”blue”, ”yellow”, ”orange” και ”purple” οι οποίες εφαρμόζουν μια παλέτα χρωμάτων που βασίζεται στο αντίστοιχο χρώμα του ονόματος της. Στην συνέχεια ο χρήστης βρίσκει διαθέσιμο ένα combo box μέσω του οποίου μπορεί να μεταβεί σε διαφορετικό προφίλ. Είναι εφικτή η δημιουργία ενός καινούριου προφίλ από τον χρήστη μετά από την πληκτρολόγιση του ονόματός του στο input field που υπάρχει στην συνέχεια. Η επιθυμία αφαίρεσης ενός προφίλ μπορεί να ικανοποιηθεί από την μετάβαση στο εκάστοτε όνομα χρήστη με την επιλογή του κουμπιού ”Delete Current Profile” να ακολουθεί για την ολοκλήρωση της διαδικασίας αυτής. Στο τέλος του πλαισίου της εφαρμογής υπάρχει μια λίστα με τα directories που έχει προσθέσει χρήστης για την εύρεση εφαρμογών από το πρόγραμμα αλλά και εγγραφές με διευθύνσεις μεμονομένων εφαρμογών στο σύστημα αρχείων. Μετά την λίστα αυτή υπάρχουν κουμπιά για την προσθήκη η αφαίρεση στοιχείων από την λίστα.

D.2.2 Καρτέλα εφαρμογών

Έχοντας εξηγήσει την λειτουργία της αρχικοποίησης της εφαρμογής και της προσαρμογής της στις ανάγκες του χρήστη μέσω της καρτέλας ρυθμίσεων μπορούμε πλέον να προχωρήσουμε στην ανάλυση της καρτέλας εφαρμογών στην οποία ο ρόλος του προγράμματος πλέον

γίνεται εμφανής μέσα από τα στοιχεία της γραφικής διεπαφής που λύνουν το πρόβλημα που έχει περιγραφεί προηγουμένως. Όπως φαίνεται, αρχικά ο χρήστης βλέπει ένα input field. Όπως υποδηλώνεται στο placeholder κείμενο του input field, σκοπός του στοιχείου αυτού είναι η αναζήτηση εφαρμογών μέσα από την λίστα εφαρμογών που είναι διαθέσιμη στην εφαρμογή. Με άλλα λόγια, ο χρήστης μπορεί να ψάξει ανάμεσα στις εφαρμογές που βλέπει η εφαρμογή μέσω των directories που δώθηκαν και των εφαρμογών που προστέθηκαν ξεχωριστά. Στην συνέχεια υπάρχει ένα combo box που τροποποιεί την ταξινόμηση των στοιχείων της λίστας με τις εφαρμογές ανάλογα με την επιλογή του χρήστη σε αύξουσα ή σε φθίνουσα σειρά. Στην συνέχεια προβάλονται οι εφαρμογές σε ξεχωριστό πλαίσιο. Ο χρήστης μπορεί να αξιοποιήσει το ποντίκι για εκτέλεση των εφαρμογών αλλά και για αναζήτηση ανοίγοντας το εικονικό πληκτρολόγιο μέσα από την καρτέλα ρυθμίσεων. Τέλος, με το δεξί κλικ πάνω στο πλαίσιο εφαρμογών εμφανίζεται ένα custom context menu μέσα από το οποίο ο χρήστης μπορεί χειροκίνητα να προσθέσει μεμονομένες εφαρμογές και να προσθέσει εφαρμογές της λίστας στην λίστα με τα αγαπημένα.

D.2.3 Καρτέλα αγαπημένων εφαρμογών

Η καρτέλα Favorites δεν έχει κάποια ουσιαστική διαφορά από την βασική καρτέλα εφαρμογών. Στην καρτέλα αυτή ο χρήστης θα βρει στοιχεία γραφικής διεπαφής που εμφανίζονται και στην πρώτη καρτέλα με την μόνη διαφορά να βρίσκεται στην λίστα εφαρμογών που είναι διαφορετική και στο context menu που πλεόν εμφανίζει μόνο επιλογή για αφαίρεση εφαρμογής από την λίστα αγαπημένων.

D.3 Εκτέλεση εφαρμογών με το πληκτρολόγιο

Σε αντίθεση με το ποντίκι, το πληκτρολόγιο έχει βασική ιδιότητα εισαγωγή κειμένου ως είσοδο στον υπολογιστή. Επομένως για την εκτέλεση των εφαρμογών μέσω αυτής της συσκευής έχει δημιουργηθεί ένα ξεχωριστό παράθυρο μέσα από το οποίο ο χρήστης έχει την δυνατότητα να πληκτρολογήσει για να αναζητήσει μέσα από την λίστα των προγραμμάτων

που γνωρίζει η εφαρμογή. Δηλαδή ή αναζήτηση γίνεται μέσα στην λίστα εφαρμογών. Με την εισαγωγή κειμένου εμφανίζονται τα αποτελέσματα αναζήτησης και στην συνέχεια ο χρήστης μπορεί να χρησιμοποιήσει τα βελάκια του πληκτρολογίου για να μεταβεί στην λίστα με τα αποτελέσματα. Τέλος, πιέζοντας το πλήκτρο Enter εκτελείται η επιλεγμένη εφαρμογή και το πρόγραμμα κρύβει το παράθυρο εκτέλεσης εφαρμογών. Η επανεμφάνιση του παραθύρου γίνεται μέσω ενός shortcut που έχει ανατεθεί και γίνεται μέσω του συνδυασμού των πλήκτρων CTRL + Space.

D.4 Εκτέλεση εφαρμογών με το χειριστήριο βιντεοπαιχνιδιών

Ένα χειριστήριο βιντεοπαιχνιδιών είναι σχεδιασμένο για τις ανάγκες μιας εμπορικής κονσόλας βιντεοπαιχνιδιών και συνήθως είναι χρήσιμο στην περιήγηση συστημάτων που είναι σχεδιασμένα για αυτήν την συσκευή. Όμως ένα λειτουργικό σύστημα όπως το Windows της Microsoft δεν παρέχει κάποια μητρική υποστήριξη τέτοιου είδους συσκευών. Επομένως χρειάζεται να προστεθεί λειτουργικότητα από την εφαρμογή για την μετάφραση δεδομένων εισόδου από το χειριστήριο ως εντολές που επιτρέπουν τον χρήστη να περιηγηθεί στο σύστημα. Για τον σκοπό αυτό αξιοποιούνται οι διάφορες συντομεύσεις που παρέχει το λειτουργικό σύστημα και υλοποιούνται συναρτήσεις που επιτρέπουν την εικονική κλήση αυτών των συντομεύσεων αλλά και άλλων εντολών εισόδου του πληκτρολογίου και του ποντικιού. Επομένως οι εντολές που στέλνονται από το χειριστήριο μεταφράζονται ως ένας συνδυασμός εντολών πληκτρολογίου και ποντικιού.

Chapter E

Επεξήγηση του κώδικα της εφαρμογής

E.1 Ο κώδικας του λογισμικού

Όπως αναφέρθηκε στην εισαγωγή το παρόν πρόγραμμα έχει γραφεί σε C++. Σκοπός είναι η δημιουργία μιας desktop εφαρμογής σε Qt6 περιβάλλον η οποία δεν θα υλοποιεί τις ανάγκες της εφαρμογής και θα αξιοποιεί τους πόρους του συστήματος για καλύτερες αποδόσεις. Στην συνέχεια θα περιγραφεί ο τρόπος με τον οποίο λειτουργεί η εφαρμογή αναφέροντας τις σημαντικότερες διαδικασίες που αναγράφονται σε κώδικα και εξηγούν πως φτάνουμε στο τελικό αποτέλεσμα που είδαμε προηγουμένως.

E.2 Η συνάρτηση main

Κάθε πρόγραμμα γραμμένο σε C++ ξεκινάει με την συνάρτηση main να αποτελεί το σημείο εκκίνησης του προγράμματος. Επομένως αξίζει να δούμε αναλυτικότερα τι συμβαίνει σε αυτό το κομμάτι του κώδικα για να μπορέσουμε να κατανοήσουμε καλύτερα την ροή και τις ενέργειες του προγράμματος.

```
int main(int argc, char* argv[])
{
    QApplication app(argc, argv);

    ApplicationExplorer explorer(app);
    explorer.setWindowTitle("P2019140 - Konstantinos Tourtsakis");
    explorer.setWindowFlags(Qt::WindowCloseButtonHint | Qt::FramelessWindowHint);
    explorer.resize(800, 600);

    // Storing screen resolution
    screen_width = GetSystemMetrics(SM_CXSCREEN);
    screen_height = GetSystemMetrics(SM_CYSCREEN);

    explorer.CreateUI();

    VirtualKeyboard QKeyboard;

    //QKeyboard.setWindowFlags(Qt::WindowCloseButtonHint | Qt::FramelessWindowHint);
    explorer.QKeyboard = &QKeyboard;


    explorer.showMaximized();
    return app.exec();
}
```

Αρχικά δημιουργείται το αντικείμενο της εφαρμογής που δέχεται τα arguments με τα οποία εκτελέστηκε. Στην συνέχεια δημιουργείται ένα αντικείμενο τύπου ApplicationExplorer. Το αντικείμενο αυτό αποτελεί το βασικό παράθυρο της εφαρμογής το οποίο κληρονομεί την κλάση QWidget και χτίζει τα αντικείμενα της γραφικής διεπαφής. Στην συνέχεια προστίθενται ορισμένα flags στο παράθυρο της εφαρμογής. Μετά αποθηκεύονται οι διαστάσεις της οθόνης του χρήστη σε global μεταβλητές οι οποίες θα χρησιμεύσουν στην προσαρμογή των διαστάσεων των στοχείων της γραφικής διεπαφής ανάλογα με τις διαστάσεις της οθόνης. Μετά ακολουθεί η δημιουργία των γραφικών στοιχείων καλώντας την μέθοδο CreateUI. Η εφαμοργή συνεχίζει με την δημιουργία ενός ακόμη αντικειμένου. Αυτήν την

φορά η κλάση QWidget κληρονομείται από την κλάση VirtualKeyboard η οποία δημιουργεί και διαχειρίζεται το παράθυρο του εικονικού πληκτρολογίου. Στην συνέχεια δημιουργείται αντίγραφο της θέσης μνήμης του αντικειμένου αυτού σε μια μεταβλητή της κλάσης του αρχικού παραθύρου με σκοπό την επικοινωνία μεταξύ των δύο παραθύρων της εφαρμογής κατά την περιήγηση του χρήστη. Τέλος γίνεται κλήση για προβολή του βασικού παραθύρου της εφαρμογής με τις μέγιστες διαστάσεις παραθύρου και μετά εκτελείται η επανάληψη της Qt6 εφαρμογής.

E.3 Η κλάση ApplicationExplorer

Μια εφαρμογή που αξιοποιεί το Qt6 περιλαμβάνει αντικείμενα διαφόρων τύπων ώστε να μπορέσει να λειτουργήσει. Όπως τα περισσότερα στοιχεία της εφαρμογής, έτσι και το παράθυρο που δημιουργείται για να φιλοξενήσει την γραφική διεπαφή της είναι ένα αντικείμενο. Στην προκειμένη περίπτωση κληρονομείται η κλάση QWidget και στην συνέχεια χρησιμοποιείται ως βάση για την υλοποίηση της εφαρμογής. Παρακάτω βλέπουμε τον ορισμό των χαρακτηριστικών της κλάσης και

E.3.1 Ο constructor

Παρακάτω βλέπουμε την υλοποίηση του constructor της εφαρμογής στον οποίο δημιουργείται το χρονόμετρο με το οποίο εκτελείται η διεργασία περιήγησης του χειριστηρίου βιντεοπαιχνιδιών.

```
class ApplicationExplorer : public QWidget
{
    ApplicationExplorer(QApplication& app, QWidget* parent = nullptr) : QWidget(parent), app
    {
        // Controller task loop - Perform task constantly
        timer = new QTimer(this);
        connect(timer, &QTimer::timeout, this, &ApplicationExplorer::TaskGamepadConnection);
        timer->start();
```

```
    }
};
```

E.3.2 Η μέθοδος LoadAppData

Η μέθοδος LoadAppData είναι υπεύθυνη για την αρχικοποίηση του προγράμματος. Τα δεδομένα που διαχειρίζεται η μέθοδος αυτή είναι ανεξάρτητα από την δραστηριότητα του χρήστη και φορτώνονται κατά την εκκίνηση της εφαρμογής. Επομένως έχει να κάνει με δεδομένα που το πρόγραμμα θα τροποποιήσει το ίδιο κατά την εκτέλεση. Ένα από αυτά είναι η αποθήκευση της κατάστασης του προγράμματος για την ανίχνευση της πρώτης εκτέλεσης στην οποία ο χρήστης λαμβάνει ένα εισαγωγικό μήνυμα.

E.3.3 Η μέθοδος SaveAppData

Η μέθοδος SaveAppData διαχειρίζεται τα ίδια δεδομένα με την LoadAppData με την μόνη διαφορά να βρίσκεται στην λειτουργία της. Όπως είναι προφανές, σκοπός της είναι η αποθήκευση των δεδομένων αυτών.

E.3.4 Η μέθοδος LoadProfile

Η μέθοδος LoadProfile επίσης αρχικοποιεί το πρόγραμμα κατά την εκκίνηση. Όμως η διαφορά με την LoadAppData είναι πως στην περίπτωση αυτή γίνεται φόρτωση δεδομένων που αφορούν την εμπειρία του κάθε χρήστη και όχι του προγράμματος. Επομένως, ενώ η μέθοδος LoadAppData φορτώνει τα ίδια δεδομένα για όλους τους χρήστες, η LoadProfile θα φορτώσει διαφορετικές τιμές ανάλογα με τις ενέργειες του κάθε χρήστη.

E.3.5 Η μέθοδος SaveProfile

Η μέθοδος SaveProfile είναι η αντίστοιχη της SaveAppData. Σκοπός της είναι η αποθήκευση των δεδομένων που διαχειρίζεται η LoadProfile.

E.3.6 Η μέθοδος OnApplicationExit

Η μέθοδος αυτή εκτελεί διαδικασίες που πρέπει να πραγματοποιηθούν κατά την έξοδο εκτέλεσης του προγράμματος. Για παράδειγμα, Η SaveProfile καλείται μέσω αυτής της μεθόδου. Η εκτέλεσή της γίνεται συνδέοντάς την εφαρμογή και ορίζοντας ως signal το QCoreApplication::aboutToQuit.

```
QObject::connect(&app, &QCoreApplication::aboutToQuit, this, &ApplicationExplorer::OnApplica
```

E.3.7 Η μέθοδος DirectoryListUpdated

Η μέθοδος αυτή αρχικά καθαρίζει την λίστα με τις γνωστές εφαρμογές. Καλείται όταν έχει γίνει κάποια αλλαγή στην λίστα με τα directories και με κάθε προσθήκη ή αφαίρεση directory προσπελαύνει την λίστα για την ενημέρωση των εφαρμογών που είναι γνωστές στο πρόγραμμα. Η λίστα με τα directories μπορεί να περιλαμβάνει διευθύνσεις σε μεμονωμένα αρχεία. Επομένως γίνεται έλεγχος. Αν το στοιχείο είναι directory τότε καλέιται η μέθοδος ExploreDirectoryFiles με την διεύθυνση του directory. Αν το στοιχείο είναι αρχείο τότε το προσθέτει στην λίστα εφαρμογών.

E.3.8 Η μέθοδος ExploreDirectoryFiles

Η μέθοδος αυτή δέχεται ως είσοδο ένα directory και στην συνέχεια προσπελαύνονται όλα τα στοιχεία που βρίσκονται σε αυτό. Αν το στοιχείο είναι ένα αρχείο τύπου ”.exe”, ”.lnk” ή ”.url” τότε η μέθοδος το προσθέτει στην λίστα εφαρμογών του χρήστη. Αν ένα στοιχείο είναι και αυτό directory τότε γίνεται αναδρομική κλήση της μεθόδου.

E.3.9 Η μέθοδος UpdateListWidget

Η μέθοδος UpdateListWidget δέχεται ως είσοδο μια string λίστα που περιλαμβάνει τις διευθύνσεις με τις εφαρμογές και μια λίστα με τα αντικείμενα που θα δημιουργηθούν για την προβολή των εφαρμογών αυτών στην γραφική διεπαφή. Κατά την διαδικασία αυτή

γίνονται ενέργειες σχετικά με την ανάκτηση του εικονιδίου της εφαρμογής και τις ρυθμίσεις του κάθε χρήστη σχετικά με την συμπερίληψη της επέκτασης του αρχείου ή όχι.

E.3.10 Η μέθοδος SearchApplication

Η μέθοδος αυτή εκτελεί αναζήτηση μέσα στις λίστες εφαρμογών. Δέχεται ως είσοδο ένα input field από το οποίο θα προέλθει η είσοδος της αναζήτησης, την λίστα αντικειμένων στην οποία προστίθενται οι εφαρμογές ως στοιχεία για προβολή στην γραφική διεπαφή και την λίστα με τις διευθύνσεις των εφαρμογών που γνωρίζει το πρόγραμμα. Αφού γίνει ενημέρωση της λίστας εφαρμογών σύμφωνα με την είσοδο της αναζήτησης, γίνεται τελικά κλήση της μεθόδου UpdateListWidget για την ενημέρωση της λίστας αντικειμένων που χρησιμοποιεί η μέθοδος αναζήτησης.

E.3.11 Η μέθοδος ExecuteApplication

Η μέθοδος αυτή δέχεται ως είσοδο ένα στοιχείως της λίστας αντικειμένων των εφαρμογών που προβάλονται στο πρόγραμμα. Αν το αντικείμενο είναι έγκυρο τότε η μέθοδος φροντίζει για την εκτέλεσή του ανάλογα με την κατάληξη της διεύθυνσης της εφαρμογής. Αμέσως μετά προσμετρά την εκτέλεση του αρχείου σε ένα QMap με τις δημοφιλείς εφαρμογές που προκύπτουν από το πλήθος των εκτελέσεων τους.

E.3.12 Η μέθοδος UpdatePopularAppsList

Η μέθοδος UpdatePopularAppsList δημιουργεί μια λίστα από το QMap που αναφέρθηκε στην μέθοδο ExecuteApplication προηγουμένως. Αυτό που κάνει είναι να ταξινομεί τα αρχεία που εκτελέστηκαν σε φθίνουσα σειρά. Στην συνέχεια αποθηκεύει τα 10 πιο δημοφιλή αρχεία σε μια ξεχωριστή string λίστα που στην συνέχεια θα αξιοποιηθεί για την δημιουργία προτάσεων κατά την αναζήτηση του χρήστη στο input field.

E.3.13 Η μέθοδος TaskGamepadConnection

Η μέθοδος TaskGamepadConnection είναι υπεύθυνη για την διαχείριση του χειριστηρίου βιντεοπαιχνιών. Μιας και το Qt μας επιτρέπει να δημιουργήσουμε διαδικασίες που εκτελούνται ανά συγκεκριμένα χρονικά διαστήματα, η λειτουργικότητα αυτή συνδυάζεται με τον ορισμό ενός μηδενικού χρονικού ορίου. Επομένως η μέθοδος αυτή καλείται συνεχώς για να γίνεται έλεγχος συνδεσιμότητας του χειριστηρίου. Η μέθοδος αρχικά κάνει έλεγχο για αρχικοποίηση του χειριστηρίου δημιουργόντας ένα αντικείμενεο τύπου XBOXController και στην συνέχεια εφόσον υπάρχει συνδεδεμένο χειριστήριο τότε καλεί μια συνάρτηση ρουτίνας με όνομα ControllerNavigation στην οποία γίνονται οι έλεγχοι για τις βασικές εντολές εισόδου που διαθέτει το χειριστήριο. Αυτές αφορούν την κλήση του εικονικού πληκτρολογίου, την απενεργοποίηση των λειτουργιών του χειριστηρίου (αποσκοπεί στην χρήση του χειριστηρίου από άλλες εφαρμογές όπως βιντεοπαιχνίδια χωρίς το πρόγραμμα να συγκρούεται με αυτές) αλλά και την κλήση μιας ακόμη συνάρτησης ρουτίνας στην οποία ελέγχονται οι υπόλοιπες εντολές εισόδου.

E.3.14 Η μέθοδος TaskHandleDevicesUICommunication

Η μέθοδος αυτή είναι υπεύθυνη για την επικοινωνία μεταξύ της εφαρμογής και δύο άλλων συσκευών εισόδου. Οι λειτουργίες αυτές δεν υλοποιούνται με το Qt, επομένως έχει πάλι δημιουργηθεί μια διαδικασία που εκτελείται συνεχώς από το Qt και κάνει ελέγχους στο παρασκήνιο για εντολές εισόδου από το πληκτρολόγιο και το χειριστήριο. Το πληκτρολόγιο στην περίπτωση αυτή επικοινωνεί με το παράθυρο αναζήτησης εφαρμογών όπου ο χρήστης μπορεί να το ενεργοποιήσει πιέζοντας LCTRL και Space. Στην συνέχεια μπορεί να γράψει κείμενο και να μεταβεί λίστα εφαρμογών με τα βελάκια. Αντίστοιχα, το χειριστήριο μπορεί στην λίστα εφαρμογών του αρχικού παραθύρου της εφαρμογής να μετακινηθεί μεταξύ των εφαρμογών και να φτάσει στο input field πατώντας το πάνω βελάκι όσο βρίσκεται στην πρώτη σειρά εφαρμογών.

E.3.15 Η μέθοδος SetupUI

Η μέθοδος SetupUI είναι από τις πρώτες μεθόδους που καλούνται από την εφαρμογή και είναι υπεύθυνη για την δημιουργία των στοιχείων της γραφικής διεπαφής που τελικά βλέπει ο χρήστης. Στην μέθοδο αυτή δημιουργούνται οι καρτέλες και τα πλαίσια στα οποία θα προστεθούν στην συνέχεια στοιχεία όπως κουμπιά, λίστες, checkboxes, input fields και combo boxes. Στην συνέχεια γίνεται σύνδεση των στοιχείων με τις μεθόδους που τους αντιστοιχούν και στο τέλος δημιουργείται επίσης το παράθυρο και η γραφική διεπαφή του παραθύρου αναζήτησης εφαρμογών πληκτρολογίου.

E.3.16 Η μέθοδος ShowCustomContextMenu

Η μέθοδος αυτή δημιουργεί ένα context menu στο επιλεγμένο στοιχείο της εφαρμογής και το εμφανίζει όταν γίνει ένα δεξί κλικ από τον χρήστη πάνω σε αυτό. Οι λειτουργίες που περιλαμβάνει είναι η χειροκίνητη προσθήκη κάποιας εφαρμογής που δεν υπάρχει σε κάποιο από τα directories που έδωσε ο χρήστης στο πρόγραμμα. Η προσθήκη αυτή γίνεται στην λίστα με τα directories όπως αναφέρθηκε προηγουμένως. Η δεύτερη ενέργεια που μπορεί να γίνει από το context menu είναι η προσθήκη/αφαίρεση μιας εφαρμογής στην λίστα των αγαπημένων εφαρμογών του χρήστη. Η λειτουργία αυτή προσαρμόζεται ανάλογα με την κατάσταση μιας εφαρμογής (αγαπημένη ή όχι).

E.3.17 Η μέθοδος SetupIntroScreen

Η μέθοδος αυτή καλείται μόνο κατά την πρώτη εκτέλεση του προγράμματος και σκοπός της είναι η δημιουργία της γραφικής διεπαφής της εισαγωγικής οθόνης στην οποία ο χρήστης καλοσορίζεται από το πρόγραμμα και καλείται να εισάγει μερικά στοιχεία για την επιτυχή ολοκλήρωση της αρχικοποίησης του προγράμματος.

E.3.18 Η μέθοδος UpdateUIPalette

Η μέθοδος UpdateUIPalette δέχεται ένα string για είσοδο και στην συνέχεια ελέγχει την τιμή του σε μία δομή επιλογής if...else if. Άνάλογα με την τιμή αυτή εφαρμόζεται η κατάλληλη θεματολογία για τα χρώματα της εφαρμόγης. Η τιμή εισόδου προέρχεται από ένα combo box που είναι διαθέσιμο στον χρήστη στην καρτέλα ρυθμίσεων.

E.4 Η κλάση VirtualKeyboard

Η κλάση VirtualKeyboard δημιουργεί το αντικείμενο του εικονικού πληκτρολογίου και διαχειρίζεται τα δεδομένα του. Μέσω της κλάσης αυτής ο χρήστης μπορεί να γράψει ένα κείμενο εισόδου και στην συνέχεια να το στείλει σε ένα input field (αν χρήσιμοποιεί χειριστήριο βιντεοπαιχνιδιών) ή να το αντιγράψει για επικόλληση μέσω του ποντικιού.

E.4.1 Ο constructor της κλάσης

Ο constructor αυτή της κλάσης αρχικοποιεί το input field στο οποίο θα εμφανιστεί το κείμενο που πληκτρολογεί ο χρήστης και στην συνέχεια καλεί την CreateKeyboardUI για το χτίσιμο της γραφικής διεπαφής.

E.4.2 Η μέθοδος GetWidgetAt

Για την ανίχνευση της θέσης στην οποία βρίσκεται χρήστης στο εικονικό πληκτρολόγιο πρέπει να ανακτηθεί το αντικείμενο το οποίο βρίσκεται στην τωρινή θέση του δείκτη. Επομένως η μέθοδος αυτή ελέγχει όλες τις θέσεις του δισδιάστατου πλαισίου στο οποίο τοποθετούνται τα εικονικά πλήκτρα, όπου στην προκειμένη περίπτωση υλοποιούνται ως κουμπιά, ανακτά το αντικείμενο που βρίσκεται στην θέση αυτή ——————————

E.4.3 Η μέθοδος SendKeyboardInput

Η μέθοδος SendKeyboardInput καλείται από το χειριστήριο για την εισαγωγή και αποστολή κειμένου από το εικονικό πληκτρολόγιο προς το ενεργό input field. Αυτό που κάνει είναι να κρατάει το τελευταίο ενεργό παράθυρο και στην συνέχεια να φέρνει στο προσκήνιο το παράθυρο του εικονικού πληκτρολογίου. Κατά την διάρκεια κλήσης της SendKeyboard-Input γίνεται έλεγχος για τα κουμπιά που πιέζονται από τον χρήστη ώστε να ενημερωθεί η γραφική διεπαφή του πληκτρολογίου και να εμφανιστεί η θέση στην οποία βρίσκεται ο χρήστης πάνω στο πληκτρολόγιο. Όταν ο χρήστης τελειώσει και πιέσει το κουμπί ”Start” του χειριστηρίου το παράθυρο του πληκτρολογίου κρύβεται και στο προσκήνιο εμφανίζεται το τελευταίο παράθυρο που κράτησε η εφαρμογή. Τέλος στέλνεται το input στο παράθυρο και εισάγεται στο input field.

E.4.4 Η μέθοδος CreateKeyboardUI

Η μέθοδος αυτή είναι υπεύθυνη για την δημιουργία της γραφικής διεπαφής του εικονικού πληκτρολογίου. Αρχικά δημιουργεί τα απαραίτητα πλαίσια και στην συνέχεια προσθέτει τα κατάλληλα κουμπιά με βάση έναν πίνακα που περιλαμβάνει το σχέδιο του πληκτρολογίου. Το σχήμα του UI είναι βασισμένο στο σχήμα πληκτρολογίου QWERTY.

E.5 Η ροή του προγράμματος

Γνωρίζοντας πλέον τα σημαντικά κομμάτια που απαρτίζουν την εφαρμογή, μπορούμε πλεόν να περιγράψουμε με ποιο τρόπο συνδυάζονται για την παραγωγή του τελικού αποτελέσματος. Για να μπορέσουμε να καταλάβουμε καλύτερα την ροή του προγράμματος θα γίνει περιγραφή με κώδικα όπου θα εξηγείται η σειρά εκτέλεσης των εντολών και των μεθόδων αγνοόντας τις λεπτομέρειες και εστιάζοντας στην πλήρη εικόνα. Ξεκινώντας λοιπόν από την συνάρτηση main έχουμε την δημιουργία του αντικειμένου ”app” της εφαρμογής. Στην συνέχεια δημιουργείται το ApplicationExplorer αντικείμενο ”περνόντας” στον con-

structor του το αντικείμενο της εφαρμογής. Αυτό θα επιτρέψει στην κλάση να διαχειριστεί την εφαρμογή και πρακτικά θα το αξιοποιήσει στην αλλαγή της παλέτας της γραφικής διεπαφής για την αλλαγή της θεματολογίας.

```
int main(int argc, char* argv[])
{
    QApplication app(argc, argv);

    ApplicationExplorer explorer(app);
    // ....



    return app.exec();
}
```

Με την κλήση του constructor έχουμε την αρχικοποίηση του χρονικού διαστήματος που ορίζεται για το χειριστήριο και συνδέεται με την μέθοδο TaskGamepadConnection.

```
ApplicationExplorer(QApplication& app, QWidget* parent = nullptr) : QWidget(parent), app(app
{
    timer = new QTimer(this);
    connect(timer, &QTimer::timeout, this, &ApplicationExplorer::TaskGamepadConnection);
    timer->start();
}
```

Επιστρέφοντας στην main, ορίζονται κάποια χαρακτηριστικά στο παράθυρο της εφαρμογής και στην συνέχεια ανακτάται η ανάλυση της οθόνης. Γνωρίζοντας πλέον την ανάλυση της οθόνης, η εφαρμογή είναι σε θέση να δημιουργήσει τα στοιχεία της γραφικής διεπαφής των οποίων το μέγεθος είναι σχετικό της ανάλυσης.

```
int main(int argc, char* argv[])
{
    QApplication app(argc, argv);

    // ...

    explorer.setWindowTitle("P2019140 - Konstantinos Tourtsakis");
```

```
    explorer.setWindowFlags(Qt::WindowCloseButtonHint | Qt::FramelessWindowHint);
    explorer.resize(800, 600);

    // Storing screen resolution
    screen_width = GetSystemMetrics(SM_CXSCREEN);
    screen_height = GetSystemMetrics(SM_CYSCREEN);
    //std::cout << "Screen resolution: " << screen_width << "x" << screen_height << std::end

    explorer.CreateUI();

    // ...


    return app.exec();
}
```

Για τον σκοπό αυτό καλείται η μέθοδος CreateUI. Στην μέθοδο αυτή γίνεται αρχικά έλεγχος για την πρώτη εκτέλεση του προγράμματος μετά την φόρτωση των δεδομένων της εφαρμογής από τον δίσκο. Αν είναι η πρώτη φορά εκτέλεσης του προγράμματος τότε το γραφικό περιβάλλον που δημιουργείται έχει στόχο την εισαγωγή και το καλωσόρισμα του χρήστη στην εφαρμογή. Με την ολοκλήρωση των απαραίτητων ενεργειών που ζητούνται από το πρόγραμμα, ο χρήστης στην συνέχεια εισέρχεται στην γραφική διεπαφή του προγράμματος. Αυτό συμβαίνει αφού κληθεί για δεύτερη φορά η CreateUI από την SetupIntroScreen όπου πλέον η εφαρμογή γνωρίζει πως δεν είναι η πρώτη εκτέλεση που συμβαίνει και καλεί την SetupUI.

```
void CreateUI()
{
    LoadAppData();

    if (is_first_launch)
    {
        SetupIntroScreen();
    }
    else
```

```
    {
        SetupUI();
    }
}
```

Πίσω στην συνάρτηση main, γίνεται δημιουργία του εικονικού πληκτρολογίου και η κλήση του constructor του. Στην συνέχεια ακολουθεί ο ορισμός κάποιων χαρακτηριστικών στο παράθυρο του πληκτρολογίου και μετά γίνεται αποθήκευση της διεύθυνσής του στο αντικείμενο της κλάσης ApplicationExplorer η οποία το διαχειρίζεται για την επικοινωνία του με την εφαρμογή.

```
int main(int argc, char* argv[])
{
    QApplication app(argc, argv);

    // ...
    VirtualKeyboard QKeyboard;

    //QKeyboard.setWindowFlags(Qt::WindowCloseButtonHint | Qt::FramelessWindowHint);
    explorer.QKeyboard = &QKeyboard;
    // ...


    return app.exec();
}
```

Ρίχνοντας μια ματιά στον constructor της VirtualKeyboard, μπορούμε να δούμε πως το πρόγραμμα ουσιαστικά συνεχίζει με την δημιουργία της γραφικής του διεπαφής.

```
VirtualKeyboard()
{
    setWindowTitle("Virtual Keyboard");
    virtual_input = new QLineEdit(this);
    virtual_input->setReadOnly(true);
    virtual_input->setPlaceholderText("Your input");
    CreateKeyboardUI();
}
```

Όπως φάινεται παρακάτω, για την δημιουργία της γραφικής διεπαφής γίνεται χρήση κουμπιών που δημιουργούνται σύμφωνα με τον πίνακα πλήκτρων. Στην συνέχεια προστίθενται μερικά ακόμη κουμπιά και τέλος γίνεται έλεγχος από τον οποίο η εφαρμογή γνωρίζει εάν το πληκτρολόγιο χρησιμοποιείται από το χειριστήριο ή από το ποντίκι, όπου στην συνέχεια φροντίζει για την προβολή διαφορετικού μηνύματος προς τις οδηγίες του χρήστη.

```
void CreateKeyboardUI()
{

    // ....

    const QStringList UpperKeyLayout
    {
        "!", "@", "#", "$", "%", "^", "&", "*", "(", ")",
        "Q", "W", "E", "R", "T", "Y", "U", "I", "O", "P",
        "A", "S", "D", "F", "G", "H", "J", "K", "L", "_",
        "Z", "X", "C", "V", "B", "N", "M", "<", ">", "?",
    };

    // ....

    for (const QString& text : KeyLayout)
    {
        QPushButton* button = new QPushButton(text);

        // ....

        layout_keyboard->addWidget(button, row, column);

        // ...
    }


    QPushButton* button_back = new QPushButton("Backspace");

    // ....
```

```
    connect(button_back, &QPushButton::clicked, this, [=]()
        {
            virtual_input->backspace();
        });

    // ....

    column = 0;
    row++;


    layout_keyboard->addWidget(button_back, row, column);

    // ....

    if (getting_input)
    {
        label_instructions = new QLabel("A: type key   Y: insert space   X: Backspace   Star
    }
    else
    {
        label_instructions = new QLabel("Click on the keys and then use the copy button to s
    }

    // ....
}
```

Επιστρέφοντας στην main, γίνεται τελικά η προβολή της γραφικής διεπαφής του προγράμματος και στην συνέχεια ξεκινάει η βασική επανάληψη της εφαρμογής που την διαχειρίζεται το Qt. Αυτό περιλαμβάνει και όλες τις διαδικασίες που εκτελούνται συνεχώς και τις οποίες θα δούμε στην συνέχεια.

```
int main(int argc, char* argv[])
{
    QApplication app(argc, argv);
```

```
    // ....

    explorer.showMaximized();
    return app.exec();
}
```

E.6 Οι επαναληπτικές διαδικασίες στο παρασκήνιο

Όπως προαναφέρθηκε, το πρόγραμμα στηρίζεται αρκετά σε διαδικασίες που εκτελούνται στο παρασκήνιο οι οποίες διαχειρίζονται εντολές εισόδου του χειριστηρίου και του πληκτρολογίου. Αυτές οι διαδικασίες καλούνται συνεχώς από το Qt και είναι απαραίτητες για την σωστή λειτουργία της εφαρμογής.

E.6.1 Η συνάρτηση GamepadInputChecks

Στο παρασκήνιο λοιπόν έχουμε την συνάρτηση GamepadInputChecks η οποία ελέγχει τις πιέσεις κουμπιών στο χειριστήριο βιντεοπαιχνιδιών. Η συνάρτηση αυτή αξιοποιεί μεθόδους της κλάσης XBOXController για τους ελέγχους και συναρτήσεις που έχουν υλοποιηθεί για την αποστολή εικονικών εντολών εισόδου προς το σύστημα. Παρακάτω βλέπουμε την κλήση της συνάρτησης SimulateKey η οποία στέλνει στο σύστημα μια εντολή εισόδου του πλήκτρου που δέχεται ως παράμετρο.

```
void GamepadInputChecks()
{
    // ....

    if (user->IsButtonJustDown(GAMEPAD_START))
    {
        SimulateKey(VK_RETURN);
    }
```

```
    // ....
}
```

Η συνάρτηση αυτή περιλαμβάνει επίσης κλήσεις συναρτήσεων που έχουν δημιουργηθεί για την αποστολή εισόδου συντομεύσεων αλλά και την μετακίνηση του κέρσορα.

```
void CursorNavigation()
{
    POINT cursor_pos;
    GetCursorPos(&cursor_pos);

    // ....

    if (lengthsq > dead_zone * dead_zone)
    {
        // Υπολογισμόςτοποθεσίαςμετακίνησηςκέρσορα
    }
    else
    {
        rest_cursor = GetTickCount() + 500;
    }

    // ....

    SetCursorPos((int)x, (int)y);
}
```

E.6.2 Η μέθοδος TaskHandleDevicesUICommunication

Επιστρέφοντας στην μέθοδο TaskHandleDevicesUICommunication, παρακάτω βλέπουμε πλέον την γενική μορφή αυτής της μεθόδου. Υπενθυμίζεται πως αυτή η μέθοδος είναι υπεύθυνη για ελέγχους εντολών εισόδου που αφορούν την επικοινωνία μεταξύ του χειριστηρίου και της εφαρμογής αλλά και τους πληκτρολογίου με το παράθυρο αναζήτησης εφαρμογών.

```
void TaskHandleDevicesUICommunication()
{
```

```
    // Handling keyboard application launcher window hardware - software communication
    if (IsKeyDown(VK_CONTROL) && IsKeyJustDown(VK_SPACE))
    {
        if (qwid_usearch_window.isVisible())
        {
            qwid_usearch_window.hide();
        }
        else
        {
            qwid_usearch_window.show();
            // ....
        }

        return;
    }

    // ....

    // Focus on the applications list with a controller
    if (user->IsButtonJustDown(GAMEPAD_DPAD_DOWN) && line_all_search->hasFocus())
    {
        list_widget->setFocus();
    }

    // ...
}
```

Chapter F

Πως να εγκαταστήσετε την εφαρμογή

F.1 Λήψη και εγκατάσταση της εφαρμογής

Για την εγκατάσταση του προγράμματος αυτού αρχικά πρέπει να γίνει λήψη του εκτελέσιμου αρχείου εγκατάστασης. Το αρχείο αυτό βρίσκεται στο αποθετήριο του προγράμματος στο οποίο φιλοξενείται και ο κώδικας. Για την εγκατάσταση πρέπει να γίνει εκτέλεση του installer. Ο installer αρχικά δίνει την δυνατότητα επιλογής γλώσσας εγκατάστασης με τις διαθέσιμες γλώσσες να είναι τα Αγγλικά και τα Ελληνικά. Στην συνέχεια γίνεται επιλογή του directory στο οποίο θα εγκατασταθεί το πρόγραμμα και επιλογή προσθήκης συντόμευσης στην επιφάνεια εργασίας ή όχι. Με την ολοκλήρωση αυτών παραθύρων, φτάνει στο τελικό στάδιο όπου αντιγράφονται τα αρχεία στον δίσκο. Σύνδεσμος αποθετηρίου: https://github.com/Konstanti

Chapter G

Σχήματα και Πίνακες

ΤΑ σχήματα είναι απαραίτητα πολλές φορές για να κατανοήσει ο αναγνώστης καλύτερα το περιεχόμενο του κειμένου μας. Μερικές φορές είναι απαραίτητα και για το κείμενό μας. Επ ουδενί όμως, δεν επιτρέπεται η αντιγραφή (scan) σχημάτων άλλων συγγραφέων που υπάρχουν σε βιβλία, επιστημονικές εργασίες κτλ.

Όπως τα σχήματα, χρήσιμοι είναι και οι πίνακες γιατί μπορεί κάποιος με εύκολο τρόπο να βρει χρήσιμη πληροφορίας, όπως π.χ. παράθεση πειραματικών αποτελεσμάτων που από τη φύση της είναι δύσκολη και ίσως δυσνόητη όταν βρίσκεται εντός του κειμένου.

G.1 Σχήματα

Στη συνέχεια θα δωθούν μερικά παραδείγμα μορφής σχημάτων.

Το “κλασικό” σχήμα είναι αυτό που απεικονίζεται στο Σχήμα G.1. Προφανώς, το σχήμα μπορεί να θέλει να είναι λίγο μικρότερο (π.χ., όπως στο Σχήμα G.2) αλλά φροντίζουμε να μην είναι το πλάτος του μικρότερο από το 1/2 του πλάτους του κειμένου. Σε κάθε περίπτωση, τα σχήματα αριθμούνται ξεχωριστά για κάθε κεφάλαιο αρχίζοντας από το 1 με πρόθεμα το νούμερο του κεφαλαίου.

Υπάρχουν περιπτώσεις που μπορεί κάποιος να θέλει δύο ίδια σχήματα δίπλα-δίπλα,

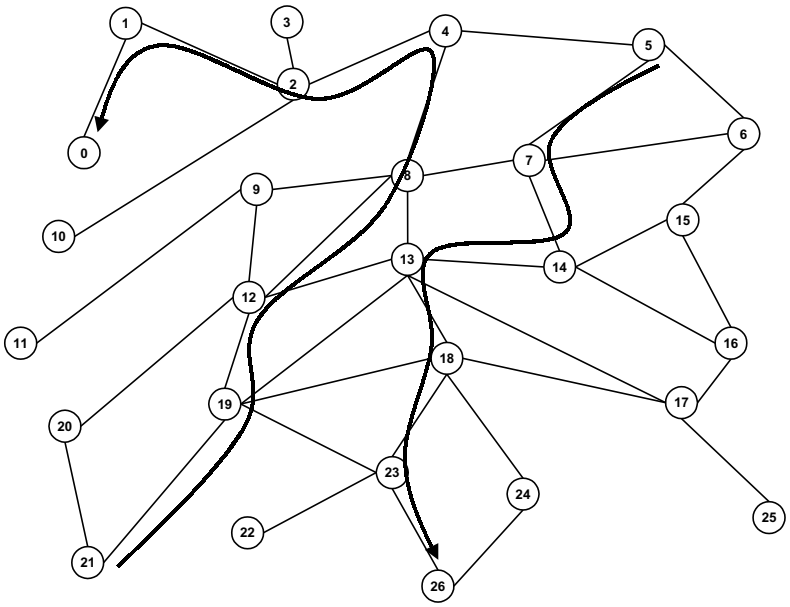


Figure G.1: Ένα δίκτυο.

όπως είναι η περίπτωση του Σχήματος G.3. Προφανώς μπορεί να γίνει ξεχωριστή αναφορά στα δύο υπο-σχήματα G.1.α και G.1.β με τον αυτό τρόπο. Το κείμενο για τα δύο υποσχήματα είναι προαιρετικό αλλά η αρίθμηση υποχρεωτική.

Όπως προηγουμένως, μπορεί να είναι επιθυμητή η αλλαγή του μεγέθους των σχημάτων, όπως φαίνεται στο Σχήμα G.4.

Τέλος, υπάρχει το ενδεχόμενο να είναι αναγκαία η παρουσίαση κάποιων σχημάτων σε τετράδα, όπως είναι το Σχήμα G.5.

Με όμοιο τρόπο όπως προηγουμένως μπορεί να επιτευχθεί αλλαγή του μεγέθους.

G.2 Πίνακες

Οι πίνακες ομοιάζουν με το κλασικό σχήμα στη μορφή και την αναφορά σε αυτούς, με μόνη διαφορά πως προηγούνται αντί να έπονται, όπως φαίνεται και στο παράδειγμα του

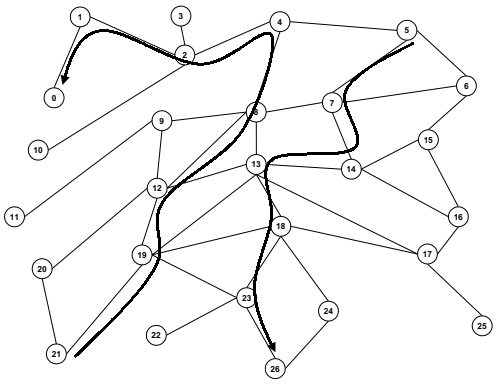
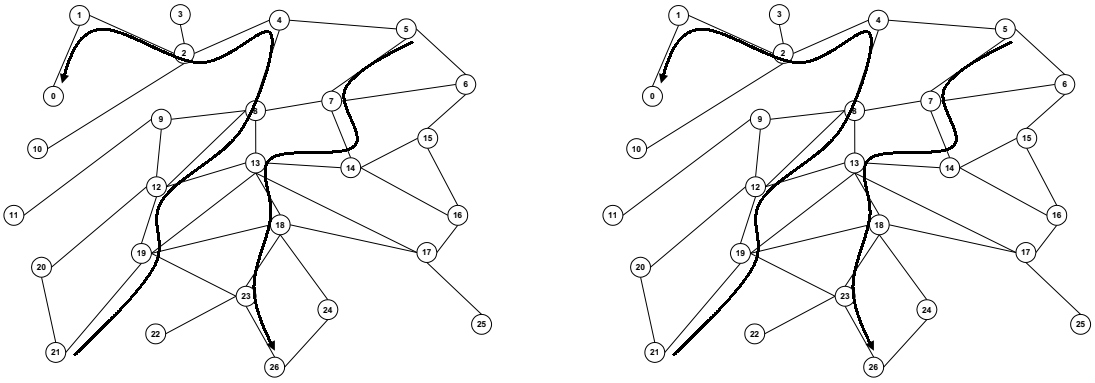


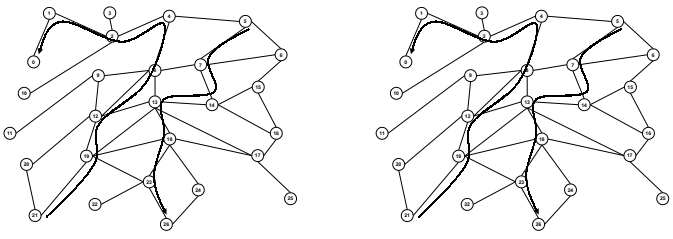
Figure G.2: Το ίδιο δίκτυο που απεικονίζεται στο Σχήμα G.1 αλλά λίγο μικρότερο.

Πίνακα G.1.



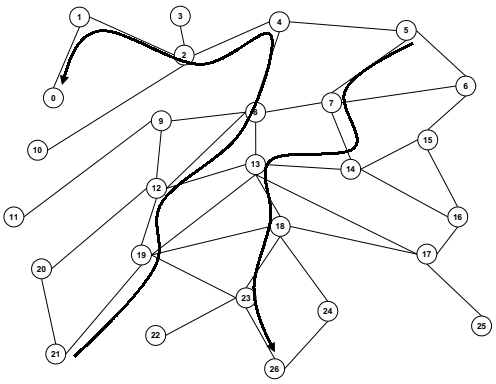
α. Κείμενο για το πρώτο. β. Κείμενο για το δεύτερο.

Figure G.3: Το ίδιο δίκτυο σε διπλή απεικόνιση.

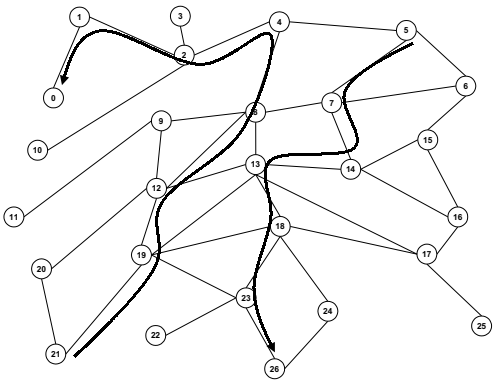


α. β.

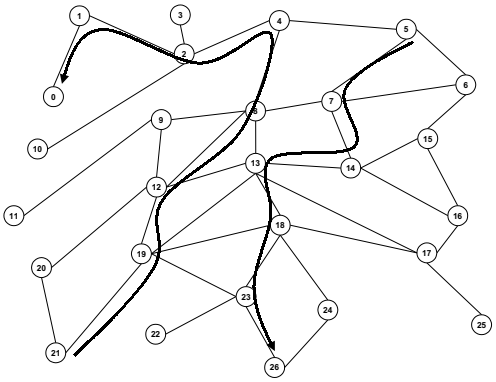
Figure G.4: Το ίδιο δίκτυο σε διπλή απεικόνιση σε σμίκρυνση και δίχως κείμενο για τα υπο-σχήματα.



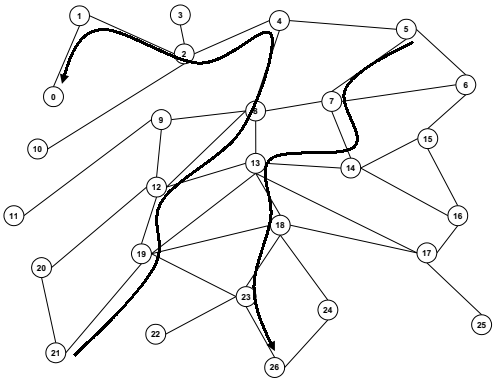
α. Κείμενο για το πρώτο.



β. Κείμενο για το δεύτερο.



γ. Κείμενο για το τρίτο.



δ. Κείμενο για το τέταρτο.

Figure G.5: Το ίδιο δίκτυο σε τετραπλή απεικόνιση.

Table G.1: Παράδειγμα Πίνακα.

	s_0	s_1	s_2	s_3	s_4	s_5	s_6
$f_0(s_\chi)$	1	2	3	4	5	6	0
$f_1(s_\chi)$	1	3	5	0	2	4	6
$f_2(s_\chi)$	4	0	3	6	2	5	1
$f_3(s_\chi)$	3	0	4	1	5	2	6
$f_4(s_\chi)$	2	5	1	4	0	3	6
$f_5(s_\chi)$	2	3	4	5	6	0	1
$f_6(s_\chi)$	6	4	2	0	5	3	1
$f_7(s_\chi)$	1	0	6	5	4	3	2
$f_8(s_\chi)$	5	6	0	1	2	3	4
$f_9(s_\chi)$	3	2	1	0	6	5	4

Chapter H

Τελευταία Μέρη

Η κύρια δομή της Πτυχιακής Εργασίας δεν μπορεί να προκαθοριστεί και ούτε θα ήταν σωστό ώστε να υπόκειται πάντα στη δημιουργικότητα του φοιτητή υπό τη συμβουλή του επιβλέποντα. Είναι όμως σημαντικό να υπάρχει ένα κεφάλαιο συμπερασμάτων στο οποίο να γίνεται τελικά σύνοψη της εργασίας και των συμπερασμάτων αυτής. Βέβαια, μια τέτοια σύνοψη έλαβε χώρα και στην Εισαγωγή. Όμως τότε ο αναγνώστης δεν είχε διαβάσει την εργασία και ο κύριος σκοπός ήταν να του προκαλέσει το ενδιαφέρον αλλά και να τον εισάγει ομαλά. Εδώ, όμως, ο σκοπός είναι να τον βοηθήσει όλα όσα διάβασε να τα καταχωρήσει στο μυαλό του ακόμα καλύτερα.

Μετά τα συμπεράσματα, και το τέλος ουσιαστικά της Πτυχιακής Εργασίας, υπάρχει ένας αριθμός από κεφάλαια τα οποία είναι χρήσιμα για τον αναγνώστη και ουσιαστικά αποτελούν το ”κερασάκι στην τούρτα.”

- Καταρχάς μπορεί να είναι κάποια επιπλέον κεφάλαια Παραρτήματος τα οποία όμως απαριθμούνται αυτόνομα (δεν έχουν συσχέτιση με την αρίθμηση των κεφαλαίων του κυρίου μέρους).
- Είναι οπωσδήποτε το κεφάλαιο της βιβλιογραφίας το οποίο αριθμεί της βιβλιογραφικές αναφορές κατά αύξοντα αριθμό πρώτης εμφάνισής τους στο κείμενο. Αν π.χ. στο σημείο αυτό γινόταν η πρώτη αναφορά τότε εν μέσω του κειμένου θα παρεμβάλλονταν

το [16]. Η βιβλιογραφία μπορεί να είναι είτε στα αγγλικά είτε στα ελληνικά ανάλογα με το προς αναφορά κείμενο.

- Το κεφάλαιο των συντμήσεων είναι ιδιαίτερα σημαντικό καθώς η Επιστήμη της Πληροφορικής έχει πολλούς ξενικούς όρους που χρησιμοποιούμε καθημερινά με συντετμημένη μορφή. Για παράδειγμα, η υπηρεσία ονομάτων περιοχής (Domain Name Services – DNS). Σε τέτοιες περιπτώσεις δίνουμε την ελληνική μετάφραση με πλάγια γράμματα και στην παρένθεση έχουμε την αγγλική έκφραση και τη σύντμηση. Στη συνέχεια είμαστε ελεύθεροι να χρησιμοποιήσουμε την ελληνική έκφραση ή τον συντμημένο αγγλικό τύπο. Προσοχή, όμως, γιατί χρησιμοποιούμε μόνο ένα από τα δύο! Το ίδιο κάνουμε και με τα ελληνικά, βάζοντας σε παρένθεση όμως μόνο τη σύντμηση. Π.χ., Οργανισμός Τηλεπικοινωνιών Ελλάδος (ΟΤΕ). Στο κεφάλαιο των συντμήσεων έχουμε διαφορετικά μέρη για τις ελληνικές και τις ξενικές συντμήσεις αλλά με αλφαβητική διάταξη.

- Το γλωσσάρι είναι ουσιαστικά η μετάφραση των ξενικών όρων που συναντάμε στο κείμενο (είτε τους χρησιμοποιούμε γιατί είθισται είτε όχι). Το γλωσσάρι δεν είναι μόνο λεξικό αλλά είναι θεμιτό να περιέχει και μια μικρή ερμηνεία του όρου.

- Τέλος, είναι σημαντικό το ευρετήριο. Στο κεφάλαιο αυτό δίνονται με αλφαβητική διάταξη οι σελίδες που απαντώνται οι σημαντικότεροι όροι μέσα στο κείμενο. Όχι κατ' ανάγκη όλοι οι σημαντικοί όροι αλλά κυρίως εκείνοι που έχουν να κάνουν με την ουσία της Πτυχιακής Εργασίας.

Παράρτημα Α'

Ενδεικτικό παράδειγμα παραρτήματος. Έπεται το κεφάλαιο της βιβλιογραφίας.

Bibliography

[1] Authors, title, information about the book, paper journal.

[2] The Qt Company Ltd, QListWidget class, Qt6 Documentation, https://doc.qt.io/qt-6/qlistwidget.html

[3] The Qt Company Ltd, QPushButton class, Qt6 Documentation, https://doc.qt.io/qt-6/qlistwidget.html

[4] The Qt Company Ltd, QLineEdit class, Qt6 Documentation, https://doc.qt.io/qt-6/qlistwidget.html

[5] The Qt Company Ltd, QIcon class, Qt6 Documentation, https://doc.qt.io/qt-6/qlistwidget.html

[6] The Qt Company Ltd, QApplication class, Qt6 Documentation, https://doc.qt.io/qt-6/qapplication.html

[7] The Qt Company Ltd, QWidget class, Qt6 Documentation, https://doc.qt.io/qt-6/qwidget.html

[8] The Qt Company Ltd, QVBoxLayout class, Qt6 Documentation, https://doc.qt.io/qt-6/qvboxlayout.html

[9] The Qt Company Ltd, QHBoxLayout class, Qt6 Documentation, https://doc.qt.io/qt-6/qhboxlayout.html

[10] The Qt Company Ltd, QGridLayout class, Qt6 Documentation, https://doc.qt.io/qt-6/qgridlayout.html

[11] The Qt Company Ltd, QDir class, Qt6 Documentation, https://doc.qt.io/qt-6/qdir.html

[12] The Qt Company Ltd, QSettings class, Qt6 Documentation, https://doc.qt.io/qt-6/qsettings.html

[13] The Qt Company Ltd, QIcon class, Qt6 Documentation, https://doc.qt.io/qt-6/qicon.html

[14] Ivan Akulov, Extracting icon using WinAPI in Qt app, Stack Overflow ερώτηση, https://stackoverflow.com/questions/12459145/extracting-icon-using-winapi-in-qt-app

[15] Microsoft, Getting Started With XInput in Windows applications, Windows Win32 Documentation, https://learn.microsoft.com/en-us/windows/win32/xinput/getting-started-with-xinput

[16] Minalien, Xbox 360 Controller Input in C++ with XInput, Code Project article, https://www.codeproject.com/articles/26949/xbox-360-controller-input-in-c-with-xinput

Συντμήσεις

DNS	Domain Name Server
UI	User Interface

Γλωσσάρι Ξενικών Όρων

Access Permissions	Δικαιώματα Πρόσβασης
Arguments	Παράμετροι
Flags	
Directory	Διευθηντήριο
Input Field	
User Interface	Γραφική Διεπαφή