Научная работа по ИКТ

на тему

“Разработка программы для изучения признаков

химических соединений”

Автор работы: Орен Сюлейман

Научный руководитель: ...

Нижний Новгород - 2023

# Оглавление

В связи с нынешними проблемами учебного процесса такими как: малая заинтересованность учеников учится, сложность восприятия изучаемой информации, отсутствием инструментов для изучения информации, остро лежит вопрос о его улучшении, модернизации.

Цель данной работы разработать инструмент для улучшения эффективности и удобства образования в области химии с помощью информационных технологий.

Для достижения данной цели решим следующий ряд задач:

* Обнаружим актуальную проблему учебного процесса.
* Разработаем инструмент для исправления данной проблемы с помощью информационных технологий.

Созданный инструмент (программа) в данной работе будет написан на языке программирования python с использованием таких библиотек как: PyQt5, random. Также при работе в некоторых местах использовалась программа QtDesigner.

Конец XX века ввёл общество в эпоху постиндустриального или информационного развития. В последние годы возможности информационных технологий для человека становятся более доступными. Это сказывается и на системе образования, и на получении знаний и умений школьников и студентов в целом. Так, IT способствуют эффективному решению различных задач в сфере обучения. Однако, далеко не все педагоги и ученики способны быстро сориентироваться в новом информационном пространстве.  
Новые информационные образовательные технологии требуют доработки и внедрению в образовательный процесс, в том числе в рамках отдельных профильных предметов.  
Также следует учитывать, что быстро развивающиеся технологии и появление различных гаджетов могут иметь как положительное, так и отрицательное воздействие на учащихся. Поэтому задача современной системы образования привлечь школьников к тем программам, приложениям и устройствам, которые облегчат процесс обучения, сделают его более доступными и интересным, а также дадут им возможность активизировать познавательную и мыслительную деятельность.  
IT-инструменты могут трансформировать и обогатить образовательные процессы, привить учащимся большую самостоятельность и ответственность по отношению к выполнению домашних заданий, мотивировать их в стремлении к новым знаниям и навыкам, не тратя на это много времени. Я считаю, что если с помощью информационных технологий получится сделать процесс освоения материала и контроля над ним более лёгким и увлекательным, то проще станет и учиться и преподавать.  
Исследования в области использования информационных образовательных технологий в системе образования ведутся достаточно давно. За это время в учебных заведениях США, Франции, Японии, России и ряда других стран было разработано множество компьютерных систем учебного назначения. Но ещё много направлений, которые можно развивать. Например, когда я готовился к ОГЭ по химии, я стал искать в интернете программу, которая помогла бы мне быстро и хорошо заучить признаки химических соединений. Такой программы я не нашёл, и решил её сделать сам. Об этом я расскажу в практической части данной работы. А сейчас я хотел бы отметить, что, оказывается, за рубежом разработку компьютерного продукта учебного назначения (методических и программно-информационных средств) считают необходимым делом в силу его высокой наукоемкости и необходимости совместной работы высококвалифицированных специалистов: психологов, преподавателей-предметников, компьютерных дизайнеров, программистов. Многие крупные зарубежные фирмы финансируют проекты создания компьютерных учебных систем в образовательных учреждениях и ведут собственные разработки в данной области. Хочется верить, что также делается или планируется делать и у нас в стране.  
Конечно же, мы давно научились пользоваться компьютерами, телефонами и другими устройствами, которые выполняют такие базовые процессы, как передача, обработка, хранение и накопление данных. А вот автоматизация знаний и способов их проверки пока что требует более глубокого изучения и новых разработок, на мой взгляд.

# Практический материал

Сначала пропишем для нашей программы начальные настройки функцией *“setupUi”*: фиксированный размер, иконку, цвет заднего фона.

Далее добавим функцию *“createElements”*, которая будет создавать различные элементы нашей программы: кнопки, поля и т.п. Пропишем три кнопки: *“btn\_OGE”*, *“btn\_EGE”*, *“btn\_All”*. Установим им размер и добавим возможность взаимодействовать с ними (нажимать). Также в этой функции создадим переменную *“stylesheetForButtons”*, в которой пропишем стили (визуальные настройки) для только что созданных и будущих кнопок. Установим стили для добавленных кнопок с помощью переменной *“stylesheetForButtons”*. Пропишем в функции *“setupUi”* запуск функции *“createElements”*, чтобы при запуске программы сразу создавались все элементы. Все последующие элементы будем создавать невидимыми, чтобы при запуске программы были видны только три начальные кнопки.

Добавим функцию *“retranslateUi”*, которая будет задавать названия для кнопок, которые будут отображатся на них, и название для нашей программы. Присвоим трём, созданным кнопкам названия “ОГЭ”, “ЕГЭ”, “Весь список”. Пропишем в функции *“setupUi”* запуск функции *“retranslateUi”*.

Добавим функцию *“addFunctions”*. Она будет привязывать к кнопкам функции, которые будут выполняться при нажатии на кнопки. Пропишем в функции *“setupUi”* запуск функции *“addFunctions”*.

Добавим функцию *“MainProtocol”*, которая будет скрывать начальные кнопки и показывать элементы тренировочного меню. В функции *“addFunctions”* привяжем функцию *“MainProtocol”* к кнопкам начального меню. Пропишем скрывание начальных кнопок.

Создадим элементы тренировочного меню. Для этого в функции *“createElements”* пропишем создание поля вывода химического соединения *“gameLabel”*, поле для ввода ответа *“requestLine”*, кнопку для проверки ответа *“btn\_go”* и кнопку для возврата к начальному меню *“btn\_Back”*. Созданным кнопкам установим стили с помощью переменной *“stylesheetForButtons”*. Создадим переменную *“stylesheetForLabelsAndLines”*, в которой пропишем стили для только что созданных и будущих полей. Установим стили для добавленных полей с помощью переменной *“stylesheetForLabelsAndLines”*. В функции *“retranslateUi”* присвоим кнопке *“btn\_go”* название “Погнали!”.

Вернёмся к функции *“MainProtocol”*. Включим возможность взаимодействовать с кнопкой для проверки ответа *“btn\_go”* и полем для ввода ответа *“requestLine”*, для последнего установим фокус, чтобы изначально было выбрано поле для ответа. Пропишем видимость для всех элементов тренировочного меню. Также очистим поле для ввода ответа.

Дальше создадим алгоритм выбора файла с соединениями и их признаками. В зависимости от нажатой на начальном меню кнопки будет выбран соответствующий файл. При нажатии на кнопку *“btn\_All”* (С надписью “Весь список”) будет выбран файл “FullProtocol.txt”, в котором находятся все собранные соединения и их признаки. Аналогично для кнопки *“btn\_OGE”* (С надписью “ОГЭ”) будет выбран файл “OGEProtocol.txt”, в котором находятся все соединения и их признаки, нужные для сдачи основного государственного экзамена, а для кнопки *“btn\_EGE”* (С надписью “ЕГЭ”) будет выбран файл “EGEProtocol.txt”, в котором находятся все соединения и их признаки, нужные для сдачи единого государственного экзамена. Далее будет случайно выбрано соединение и показано в поле для вывода химического соединения *“gameLabel”*.

Теперь добавим функцию *“checkRequest”*, которая будет проверять ответ от пользователя из поля для ввода ответа *“requestLine”*. Для начала уберём фокус с поля для ввода ответа *“requestLine”*. Далее функция будет проверять правильность ответа. Она будет брать ответ от пользователя и сравнивать с правильным ответом из соответствующего списка соединений и их признаков. Далее будет выводится итоговое меню, которое будет немного отличатся в зависимости от того, совпали ли ответы или нет. В функции *“addFunctions”* привяжем функцию *“checkRequest”* к кнопке проверки ответа *“btn\_go”*.

Создадим элементы итогового меню. Для этого в функции *“createElements”* пропишем создание поддерживающего поля *“solaceLabel”*, в котором будет выводится поддерживающие слова, поля с правильным ответом *“correctAnswerLabel”*, кнопка для повтора тренировочного цикла *“btn\_retry”*. Созданным кнопкам установим стили с помощью переменной *“stylesheetForButtons”*. Установим стили для добавленных полей с помощью переменной *“stylesheetForLabelsAndLines”*. В функции *“retranslateUi”* присвоим кнопке *“btn\_Back”* название “< Назад”.

И так в случае, когда ответы совпали, будет показываться поддерживающее поле *“solaceLabel”* с надписью “Всё верно! Молодец!”, будет выключена возможность взаимодействовать с полем для ввода ответа *“requestLine”* и кнопкой для проверки ответа *“btn\_go”*, будет показываться кнопка для повтора тренировочного цикла *“btn\_retry”*. Если же ответы не совпадут, то будут показыватся поддерживающее поле *“solaceLabel”* с надписью “Почти верно. Не переживай, ты ещё учишься!”, поле с правильным ответом *“correctAnswerLabel”*, будут скрыты поле для ввода ответа *“requestLine”* и кнопка для проверки ответа *“btn\_go”*, будет показываться кнопка для повтора тренировочного цикла *“btn\_retry”*.

Добавим функцию *“retry”*, которая будет повторять тренировочный цикл. Для начала скроем все элементы итогового меню. Дальше будет выполняться функция *“MainProtocol”* с последним выбранным списком соединений и их признаков. В функции *“addFunctions”* привяжем функцию *“retry”* к кнопке для повтора тренировочного цикла *“btn\_retry”*.

Добавим функцию *“BackToMainMenu”*, которая будет возвращать пользователя к начальному меню. Для начала скроем кнопку для возврата к начальному меню *“btn\_Back”* и поле для вывода химического соединения *“gameLabel”*. Далее будут показываться элементы начального меню. При необходимости так же будут скрыты поля *“requestLine”*, *“solaceLabel”*, *“correctAnswerLabel”* и кнопки *“btn\_go”*, *“btn\_retry”*. В функции *“addFunctions”* привяжем функцию *“BackToMainMenu”* к кнопке для возврата к начальному меню *“btn\_Back”.*

В завершении создадим файлы “FullProtocol.txt”, “OGEProtocol.txt”, “EGEProtocol.txt”, запишем туда необходимые соединения и их признаки, создадим исполняемый файл “ChemistryGame.exe”, для запуска нашей программы.

# Приложение 1.

<https://github.com/OrenSuleyman07/ChemistryGame.git>

# -\*- coding: utf-8 -\*-

from random import randrange

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtGui import QFont, QKeySequence

import keyboard

class Ui\_mainWindow(object):

def setupUi(self, mainWindow):

mainWindow.setObjectName("mainWindow")

mainWindow.setWindowModality(QtCore.Qt.NonModal)

mainWindow.setEnabled(True)

# set size

mainWindow.resize(400, 400)

mainWindow.setFixedSize(400,400)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(mainWindow.sizePolicy().hasHeightForWidth())

mainWindow.setSizePolicy(sizePolicy)

# set icon

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap("ChemistryIcon.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

mainWindow.setWindowIcon(icon)

mainWindow.setLayoutDirection(QtCore.Qt.LeftToRight)

mainWindow.setAutoFillBackground(False)

mainWindow.setStyleSheet('background-color: #202020;')

self.centralwidget = QtWidgets.QWidget(mainWindow)

self.centralwidget.setObjectName("centralwidget")

mainWindow.setCentralWidget(self.centralwidget)

self.actionprecipitation\_colors = QtWidgets.QAction(mainWindow)

self.actionprecipitation\_colors.setObjectName("actionprecipitation\_colors")

self.createElements()

self.retranslateUi(mainWindow)

QtCore.QMetaObject.connectSlotsByName(mainWindow)

self.addFunctions()

def createElements(self):

#creating main buttons

self.btn\_OGE = QtWidgets.QPushButton(self.centralwidget)

self.btn\_OGE.setEnabled(True)

self.btn\_OGE.setGeometry(QtCore.QRect(70, 150, 120, 51))

self.btn\_OGE.setObjectName("btn\_OGE")

self.btn\_EGE = QtWidgets.QPushButton(self.centralwidget)

self.btn\_EGE.setEnabled(True)

self.btn\_EGE.setGeometry(QtCore.QRect(210, 150, 120, 51))

self.btn\_EGE.setObjectName("btn\_EGE")

self.btn\_All = QtWidgets.QPushButton(self.centralwidget)

self.btn\_All.setEnabled(True)

self.btn\_All.setGeometry(QtCore.QRect(120, 220, 160, 51))

self.btn\_All.setObjectName("btn\_All")

#creating back button

self.btn\_Back = QtWidgets.QPushButton(self.centralwidget)

self.btn\_Back.setEnabled(True)

self.btn\_Back.setGeometry(QtCore.QRect(20, 350, 101, 31))

self.btn\_Back.setObjectName("btn\_Back")

self.btn\_Back.hide()

#creating game menu

self.requestLine = QtWidgets.QLineEdit(self.centralwidget)

self.requestLine.setGeometry(QtCore.QRect(10, 130, 261, 30))

self.requestLine.setObjectName("requestLine")

self.btn\_go = QtWidgets.QPushButton(self.centralwidget)

self.btn\_go.setGeometry(QtCore.QRect(280, 130, 111, 30))

self.btn\_go.setObjectName("btn\_go")

self.gameLabel = QtWidgets.QLabel(self.centralwidget)

self.gameLabel.setGeometry(QtCore.QRect(125, 60, 150, 50))

self.gameLabel.setObjectName("gameLabel")

self.correctAnswerLabel = QtWidgets.QLabel(self.centralwidget)

self.correctAnswerLabel.setGeometry(QtCore.QRect(10, 130, 381, 30))

self.correctAnswerLabel.setObjectName("correctAnswerLabel")

self.solaceLabel = QtWidgets.QLabel(self.centralwidget)

self.solaceLabel.setGeometry(QtCore.QRect(10, 180, 381, 30))

self.solaceLabel.setObjectName("solaceLabel")

self.btn\_retry = QtWidgets.QPushButton(self.centralwidget)

self.btn\_retry.setGeometry(QtCore.QRect(145, 230, 111, 30))

self.btn\_retry.setObjectName("btn\_retry")

self.requestLine.hide()

self.btn\_go.hide()

self.gameLabel.hide()

self.correctAnswerLabel.hide()

self.solaceLabel.hide()

self.btn\_retry.hide()

#style

#QtGui.QFontDatabase.addApplicationFont('OldSoviet.otf')

stylesheetForButtons = """

QPushButton{

background-color: #3B3B3B;

border: none;

border: 1px solid #303030;

border-top: 1px solid #353535;

border-radius: 7px;

color: #E1E1E1;

font-size: 15px;

}

QPushButton::hover{background-color: #323232;color: #C8C8C8;}

"""

stylesheetForLabelsAndLines = """

background-color: #3B3B3B;

border: 1px solid #303030;

border-radius: 10px;

color: #E1E1E1;

font-size: 15px;

padding: 5px;

"""

self.btn\_OGE.setStyleSheet(stylesheetForButtons)

self.btn\_EGE.setStyleSheet(stylesheetForButtons)

self.btn\_All.setStyleSheet(stylesheetForButtons)

self.btn\_Back.setStyleSheet("""

QPushButton{

background-color: #604040;

border: none;

border: 1px solid #5A3C3C;

border-top: 1px solid #644343;

border-radius: 5px;

color: #E1E1E1;

font-size: 15px;

padding-right: 10px;

}

QPushButton::hover{background-color: #553939;color: #C8C8C8;}

""")

self.requestLine.setStyleSheet(stylesheetForLabelsAndLines)

self.btn\_go.setStyleSheet(stylesheetForButtons)

self.gameLabel.setStyleSheet(stylesheetForLabelsAndLines + 'font-size: 20px;')

self.correctAnswerLabel.setStyleSheet(stylesheetForLabelsAndLines)

self.solaceLabel.setStyleSheet(stylesheetForLabelsAndLines)

self.btn\_retry.setStyleSheet(stylesheetForButtons)

#create shortcuts

# self.shortcut\_open = QtWidgets.QShortcut(QKeySequence('ENTER', self))

# self.pushButtonGood.clicked.connect(self.setFocus())

def retranslateUi(self, mainWindow):

\_translate = QtCore.QCoreApplication.translate

mainWindow.setWindowTitle(\_translate("mainWindow", "ChemistyGames"))

mainWindow.setWhatsThis(\_translate("mainWindow", "<html><head/><body><p>It\'s Game</p></body></html>"))

self.btn\_OGE.setText(\_translate("mainWindow", "ОГЭ"))

self.btn\_EGE.setText(\_translate("mainWindow", "ЕГЭ"))

self.btn\_All.setText(\_translate("mainWindow", "Весь список"))

self.btn\_Back.setText(\_translate("mainWindow", "< Назад"))

self.btn\_go.setText(\_translate("mainWindow", "Погнали!"))

self.btn\_retry.setText(\_translate("mainWindow", "Ещё раз!?"))

# self.gameLabel.setText(\_translate("mainWindow", ""))

self.actionprecipitation\_colors.setText(\_translate("mainWindow", "precipitation\_colors"))

def addFunctions(self):

self.btn\_All.clicked.connect(lambda: self.MainProtocol(self.btn\_All.objectName()))

self.btn\_OGE.clicked.connect(lambda: self.MainProtocol(self.btn\_OGE.objectName()))

self.btn\_EGE.clicked.connect(lambda: self.MainProtocol(self.btn\_EGE.objectName()))

self.btn\_Back.clicked.connect(self.BackToMainMenu)

self.btn\_go.clicked.connect(self.checkRequest)

self.btn\_retry.clicked.connect(self.retry)

def BackToMainMenu(self):

# keyboard.remove\_hotkey('enter')

self.btn\_Back.hide()

self.gameLabel.hide()

self.btn\_All.show()

self.btn\_OGE.show()

self.btn\_EGE.show()

if not self.requestLine.isHidden():

self.requestLine.hide()

self.btn\_go.hide()

if not self.solaceLabel.isHidden():

self.solaceLabel.hide()

self.btn\_retry.hide()

self.correctAnswerLabel.hide()

def MainProtocol(self, protocol\_variant):

self.last\_protocol\_variant = protocol\_variant

# keyboard.add\_hotkey('enter', self.checkRequest)

#hiding buttons

self.btn\_All.hide()

self.btn\_OGE.hide()

self.btn\_EGE.hide()

#showing back button and game menu

self.requestLine.setFocus(True)

self.requestLine.setEnabled(True)

self.btn\_go.setEnabled(True)

self.requestLine.clear()

self.btn\_Back.show()

self.requestLine.show()

self.btn\_go.show()

self.gameLabel.show()

#creating protocol\_file\_list

if protocol\_variant == 'btn\_All':

with open('FullProtocol.txt', 'rt', encoding="utf-8") as f:

protocol\_file = f.read()

self.protocol\_file\_list = protocol\_file.split('\n')

elif protocol\_variant == 'btn\_OGE':

with open('OGEProtocol.txt', 'rt', encoding="utf-8") as f:

protocol\_file = f.read()

self.protocol\_file\_list = protocol\_file.split('\n')

elif protocol\_variant == 'btn\_EGE':

with open('EGEProtocol.txt', 'rt', encoding="utf-8") as f:

protocol\_file = f.read()

self.protocol\_file\_list = protocol\_file.split('\n')

#collect and print precipitation

self.index = randrange(0, len(self.protocol\_file\_list)-1, 2)

self.gameLabel.setText(self.protocol\_file\_list[self.index])

def checkRequest(self):

# keyboard.remove\_hotkey('enter')

self.requestLine.setFocus(False)

if self.requestLine.text().lower() == self.protocol\_file\_list[self.index + 1].lower():

self.solaceLabel.setText('Всё верно! Молодец!')

self.requestLine.setEnabled(False)

self.btn\_go.setEnabled(False)

else:

self.solaceLabel.setText('Почти верно. Не переживай, ты ещё учишься!')

self.correctAnswerLabel.setText('Правильный ответ: ' + self.protocol\_file\_list[self.index + 1])

self.correctAnswerLabel.show()

self.requestLine.hide()

self.btn\_go.hide()

self.solaceLabel.show()

self.btn\_retry.show()

# keyboard.add\_hotkey('enter', self.retry)

def retry(self):

# keyboard.remove\_hotkey('enter')

self.solaceLabel.hide()

self.btn\_retry.hide()

self.correctAnswerLabel.hide()

# self.requestLine.clear()

# self.requestLine.setEnabled(True)

# self.btn\_go.setEnabled(True)

self.MainProtocol(self.last\_protocol\_variant)

# keyboard.add\_hotkey('enter', self.checkRequest)

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

mainWindow = QtWidgets.QMainWindow()

ui = Ui\_mainWindow()

ui.setupUi(mainWindow)

mainWindow.show()

sys.exit(app.exec\_())