|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |
| Институт информационных технологий | |
| Кафедра корпоративных информационных систем  **КУРСОВАЯ РАБОТА**  по дисциплине  Структура и алгоритмы обработки данных  **Тема курсовой работы**: «Создание декларативного языка для описания GUI с автоматической генерацией кода для GUI-библиотек» | |
| Студент группы ИКБО-07-18 | Басыров Сергей Амирович |
|  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  |  |
| Руководитель курсовой работы | Советов Пётр Николаевич |
|  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя) |
|  |  |
| Работа представлена к защите | «20» декабря 2019 г |
|  |  |
| Допущен к защите | «23» декабря 2019 г. |

# содержание

# ВВедение

Предметной областью данного курсового проекта является создание графического интерфейса пользователя с использованием GUI-библиотеки в рамках какого-либо высокоуровневого языка программирования.

Основные задачи, возникающие при создании GUI как правило являются: изучение GUI-библиотеки и реализация графического интерфейса. Первая задача как правило реализуема в относительно короткие сроки, однако, вторая задача может потребовать значительных временных затрат, ввиду специфики языка программирования. Если такое сравнение уместно, то например, реализация сложного графического интерфейса на базе Qt в С++ будет быстрее и не трудоёмкой задачей, нежели на базе Tkinter в Python.

Исходя из темы курсовой работы поставим вопрос, возможно ли создать декларативный язык для описания графического интерфейса, который одновременной решал бы и первую задачу и вторую, за короткие сроки? Чтобы время на его изучение было минимальным и при этом реализация графического интерфейса не уступала бы библиотечным аналогам.

Одним из вариантов решения является создание языка на базе существующей GUI-библиотеки. Под этим подразумевается создание предметно-ориентированного языка декларативного стиля решающий задачу source-to-source генерации. Иначе говоря, мы описываем графический интерфейс на понятном для разработчика языке, а получаем сгенерированный код под GUI-библиотеку. Для решения такой задачи необходимо:

1. описать язык и его особенности;
2. спроектировать и разработать транслятор;
3. спроектировать и разработать генератор кода;
4. спроектировать и разработать консольное приложение, которое позволит получать сгенерированный код.

Для реализации транслятора, генератора и приложения, в рамках курсового проекта был выбран язык Python, по требованию технического задания.

В качестве GUI-библиотеки, для которой будет генерироваться код, был выбран Tkinter, так как данная библиотека является «визитной карточкой» Python для создания графических приложений и входит в стандартную библиотеку языка.

Создаваемый декларативный язык для описания графического интерфейса будет назван Lui. Название в свою очередь является сокращением предложения – «Language of the user interface».

В итоге целью курсового проекта является автоматизация создания графического интерфейса с помощью декларативного языка. А непосредственно актуальность заключается в том, что разработчик, не знающий GUI-библиотеку Tkinter сможет для своего приложения без графического интерфейса реализовать графическую часть, в короткие сроки.

# основная часть

## 1 Описание декларативного языка для описания GUI

Перед тем как приступить к описанию языка, следует ознакомиться с существующими решениями. Однако, в большинстве случаев для описания GUI используется XML-подобная структура, что на мой взгляд кажется достаточно многословным. В результате найдено было два языка:

1. CSS – Cascade Style Sheets;
2. QML – Qt Modeling Language.

Первый язык является формальным описанием таблиц стилей, которые применяются в вёрстке веб-страниц и позволяет:

* задавать различные стили оформления для элементов;
* задавать различные стили оформления на различные события;
* создавать анимацию и трансформации содержимого;
* размечать элементы по сетке, строкам, столбцам;
* размечать положение элементов.

Второй язык является декларативным языком программирования, основанный на JavaScript. Он используется для описания GUI в рамках технологии Qt Quick и имеет обширные возможности:

* разметка GUI;
* создание адаптивного GUI;
* создание анимации;
* вставка JavaScript-кода;
* связывание событий с С++ логикой.

После анализа возможностей и синтаксиса обоих языков, следует ориентироваться на них, по части синтаксиса. В таблице 1 приведены пример кода CSS и QML.

Таблица 1 – Примеры кода на CSS и QML

|  |  |
| --- | --- |
| **CSS** | **QML** |
| .content {  width: 500px;  height: 500px;  }  .conent a:hover {  font-size: 32px;  color: red;  } | [Rectangle](https://doc.qt.io/qt-5/qml-qtquick-rectangle.html) {  width: 200  height: 100  color: "red"  [Text](https://doc.qt.io/qt-5/qml-qtquick-text.html) {  anchors.centerIn: parent  text: "Hello, World!"  }  } |

При описании языка перед нами возникают следующие задачи:

1. описание синтаксиса;
2. описание ключевых слов;
3. описание РБНФ.

## 1.1 Описание синтаксиса

Опираясь на структура кода CSS и QML, пример кода языка Lui будет выглядеть следующим образом (Таблица 2). При этом сразу же определяется базовая структура того, как будет описываться GUI.

Таблица 2 – Пример кода на языке Lui

|  |
| --- |
| **Пример кода языка Lui** |
| Window {  title: "Window app"  width: 200  height: 200    Label {  caption: "It's a label caption"  }  } |

Из примера видно, что структура кода идентична структуре кода QML. Общее описание структуры код будет выглядеть следующем образом (таблица 3). Изначально описывается Window (главный компонент), затем его свойства, после чего внутри него описывается компонент Label (надпись) и для него также указываются свойства. При этом несколько компонентов внутри Window могут идти подряд или сгруппированы в один общий компонент, отвечающий за разметку, например Grid. Соответственно то что указано в квадратных скобках может повторяться несколько раз.

Таблица 3 – Структура кода языка Lui

|  |
| --- |
| **Структура кода языка Lui** |
| Window {  title: "Window app"  [property: value]  Label {  caption: "It's a label caption"  }    [Component: { [property: value] }]  } |

Такая структура удобна для восприятия и не должна вызывать трудности в её понимании. Так как это понимается чуть ли не интуитивно. Вложенная структура друг за другом следующих элементов и свойства, характеризующие тот или иной компонент.

## 1.2 Описание ключевых слов

## 1.3 Описание РБНФ

РБНФ (Расширенные формы Бэкуса-Наура) – это форма записи грамматики контекстно-свободных грамматик для последующей их реализации в трансляторе с помощью метода рекурсивного спуска (приведённое мною определение РБНФ не является полным).

При описании синтаксиса языка мы сразу же и описали его структура в разделе 1.1 и сразу можно было сказать, что приведённое структура кода является рекурсивной. Для более точного описания данной структуры мы и воспользуемся РБНФ (таблица 4).

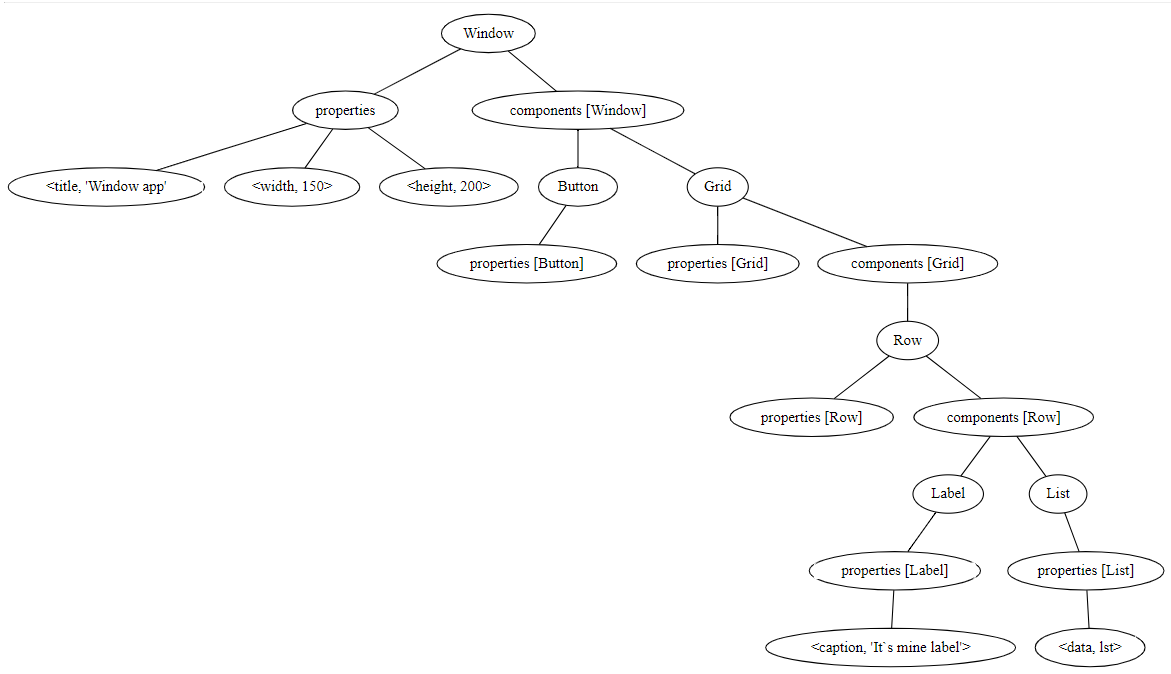
|  |  |
| --- | --- |
| **№** | **Синтаксическое уравнение** |
| 1 | syntax = mainComponent "{" {property} {component} "}" |
| 2 | component = componentName "{" {property} {component} "}" |
| 3 | property = propertyName":" string | number |
| 4 | componentName = string | {string} |
| 5 | propertyName = string | {string} |
| 6 | mainComponent = "Window" |
| 7 | string = char | {char} |
| 8 | number = digit | {digit} |
| 9 | char = "A" | ... | "Z" |
| 10 | digit = "0" | ... | "9" |

## 2 Изучение абстрактного синтаксического дерева

Пример общего синтаксического дерева



Пример синтаксического дерева для вышеприведённого кода будет



## 3 Проектирование и реализация транслятора

## 4 Изучение GUI-библиотеки Tkinter

## 5 Проектирование и реализация генератора

## 6 Проектирование и реализация приложения

В качестве итого приложения будет исполняемый файл, который принимает команды. Описание команд приведено в таблице ниже.

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| --help | Выводит справку по командам |
| --file=[filename] | Путь до файла \*.lui файла |
| --debug | Включает режим отладки, записывая все шаги в файл |
| --version | Выводит текущую версию приложения |

# заключение

В разделе статистики репозитория оказывается довольно много просмотров и даже клонов репозитория. Хм, кому же он может быть интересен?



# список используемых источников

Нормативные документы:

ГОСТ 7.32-2017

Приказ о курсовом проектировании

Книги:

Н. Вирт "Построение компиляторов"

Н. Вирт "Алгоритмы и структуры данных"

Электронные ресурсы:

<https://en.wikipedia.org/wiki/Widget_(GUI)>

<http://effbot.org/tkinterbook/>

<https://docs.python.org/3.7/library/tkinter.html>

<https://docs.python.org/3.7/library/unittest.html>

<https://docs.python.org/3.7/library/pydoc.html>