Муниципальное бюджетное общеобразовательное учреждение

Городская гимназия

***Исследовательская работа***

***«Игра Змейка на PyQt5»***

Выполнил:

Ученик 10 «Б» класса

Шигапов Камиль

Руководитель:

Учитель информатики

Садкова Ю.П.

Димитровград, 2022

1. **Введение.**

**Цель и задачи проекта:**

**Цель:**Использовать свои умения в области программирования на языке python и направить их на создание игры “Змейка”. Понять, насколько тяжело создавать и писать код для игры.

**Задачи:**

1. Разобраться в строении кода.
2. Сделать план.
3. Написать код

**Гипотеза:** Сложно ли написать игру на библиотеке PyQt 5?

**Актуальность:** Современные дети очень быстро развиваются и в два года они уже неплохо разбираются в родительских гаджетах, а к первому классу готовы сконструировать собственного двигающегося робота. Детей все больше увлекает интернет и программирование. В погоне за прогрессом, иногда полезно остановиться, вспомнить историю, задуматься о будущем, чтобы

с новыми силами вновь окунуться в работу, осваивать, а может быть и создавать новые технологии.

Актуальность данной темы, с моей точки зрения, обусловлена тем, что прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов – языков программирования.

Стоит понимать, что программирование – это всегда небольшой шаг вперед. Многие считают, что за ним будущее, и с ними нельзя не согласиться…

Основная часть.

Для программ, которые не предполагают того, что их будет использовать неподготовленный к работе с командной строкой пользователь, текстового интерфейса хватает. Существует большое количество консольных утилит, предназначенных для программистов или системных администраторов, но часто в жизни разработчика наступает момент, когда разработанную программу надо передать незнакомому с консолью пользователю. Графический интерфейс (GUI — Graphical User Interface) более «дружелюбный» к пользователям, а если в программе необходимо отображать не только текст, но и графическую или мультимедийную информацию, его использование становится необходимостью.

Для языка программирования Python есть много способов создания приложений с графическим интерфейсом, в частности, уже знакомая мне библиотека tkinter. Она используется в большом числе кроссплатформенных приложений, написанных на Python. Я абсолютно ничего не имею против tkinter, но решил рассмотреть библиотеку PyQT, так как ее возможности значительно богаче. Кроме того, на схеме Событие → Обработчик → Действие в том или ином виде построены почти все современные библиотеки, предназначенные для взаимодействия с пользователем.

Что же такое PyQT? QТ - это написанная на C++ библиотека с классами для создания графического интерфейса. Библиотека получилась настолько удачной, начала собирать вокруг себя большое сообщество программистов, которые разрабатывали приложения не только на C++, но и на других языках программирования. Это привело к тому, что и для других языков программирования стали появляться свои библиотеки-«обертки» для QT. Для Python это PyQT.

**План**

С самого начала я понимал, что хотел бы видеть в своей игре и какие эмоции вызывать у играющего. Все должно быть в пикселях и не до конца закрашенная голова змейки. В общем всё то, что было в игре “Змейка” 2005 года. А именно, бонусы, увеличивающие или замедляющие скорость змейки, ну и, конечно, яблоки, помогающие змейке рости.

**Написание кода**

За каркас я взял базовые настройки каркаса игры. В классе Window прописано всё, что отвечает за виджеты и в целом экрана(размер экрана, название окна, кнопки старта, паузы, рестарта). В терминологии PyQT (и достаточно большого числа других библиотек создания GUI) все графические приложения состоят из виджетов. В библиотеке PyQT5 существует множество модулей, но чаще других используется **QtWidgets**. Именно в нем находятся классы, соответствующие различным элементам интерфейса. Далее мы дадим им конкретное значение в зависимости от события пользователя.

Главным классом кода является Board, именно в ней прописаны основные действия змейки, бонусов, яблок, само строение змейки. Я решил использовать списки в качестве построения стенок после врезания которых змейка умирает, таким образом я написал 3 уровня после определенно съеденных яблок включается следующий уровень, не став подключать открытие следующего окна, стенки сразу появляются на данном холсте. Цвет стенок никак не связан со списком, по которым он строится, следовательно, мне пришлось вручную рисовать цвет стенок, иначе стенки были невидимы. Рисуя цвет на холсте была некоторая погрешность в пикселях поэтому на один пиксель змейка может налезать на стенку. Создание кубика: из кубиков состоит все поле, кроме стенок, они прописаны по уровням по координатам в \_\_init\_\_, draw\_stenyi\_1ur, draw\_stenyi\_2ur

Я не просто так в самом начале подключил библиотеку random, благодаря ему разные бонусы и еду есть список, в который закидываются различные значения, рандомайзер ставит объекты в разные места.

С каждым отрезком времени у змейки меняются координаты и у объектов так же имеются координаты. В функциях is\_food\_collision, Is\_Bad\_bonus\_collision, is\_good\_bonus\_collision идёт проверка съела ли змейка еду.

Основные моменты кода я прописал и теперь можно начинать считывать события. Важнее узнать, что кнопка мыши нажалась, чем получить информацию о том, что она нажата.

Любая игра также управляется событиями. Что же это за события?

Прежде всего, это события пользовательского ввода: игрок нажал клавишу на клавиатуре, подвинул мышь, нажал на кнопку закрытия окна и т. д. На каждом шаге главного игрового цикла мы разбираем накопившиеся события.

Несмотря на то, что цикл работает очень быстро, за одну итерацию наступивших событий может быть несколько. Поэтому в программе появляется второй внутренний цикл, который обрабатывает все произошедшие события (разбирает очередь событий).

PyQT не славится конечно своими игровыми событиями, для этого больше всего подойдёт библиотека PyGame. Но раз мы уже начали писать код, то вывернемя, но проверим события.

Функция Move\_snake передвигает змейку, считывая нажатия с клавиш keyPressEvent.

Цвет змейки и бонусов прописан в функциях draw\_snake\_square, draw\_food\_square, draw\_bad\_bonus\_square, draw\_bonus\_good\_square.

Необходимо зациклить процесс проверки съела ли змейка еду, в положительном случае последняя координата относительно оси ох и оу прибавляется блок.

Если же это бонус, то возникает ускорение или замедление visok\_speed, medl\_speed. Функция is\_suicide проверяет условие, врезалась ли змейка в себя.

И для настоящих молодцов, прошедших игру, написана функция win\_game, по которой вылезает окно выигрыша!!!

На протяжении всего проекта велся счетчик каждой функции, помогающий работе программы, прописанная ещё в начале кода.

**ЗАКЛЮЧЕНИЕ**

По итогу, я получил полностью рабочий код, выполняющий соответствующие требования пользователя. Помимо всего это не просто код, а игровой код, что в основном не предназначено для игр, т.е. мою гипотеза полностью оправдалась.

Я повторил базовые элементы языка python. В проекте были использованы различные библиотеки PyQt, списки, их методы, рандом, функции, классы.

Список литературы

1) Учебники Лицея Академии Яндекса

2) https://russianblogs.com/article/56962160995/

3) https://storedigital.ru/2019/07/28/biblioteka-python-rabota-s-pyqt/