

МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЁЖИ РЕСПУБЛИКИ КРЫМ

ГБОУВО РК «КРЫМСКИЙ ИНЖЕНЕРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ФЕВЗИ ЯКУБОВА»

Факультет экономики, менеджмента и информационных технологий

Кафедра прикладной информатики

Направление подготовки 09.03.03 Прикладная информатика

Профиль «Прикладная информатика в информационной сфере»

***КУРСОВОЙ ПРОЕКТ***

По дисциплине «Объектно-ориентированное программирование»

на тему:

**«Разработка аркадной 2D игры»**

Студента III курса

группы И-1-17

очной формы обучения

Близнюк М.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Научный руководитель:

Ст.пр. Сейдаметов Г.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Симферополь – 2020 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc43112814)

[ГЛАВА 1. Аналитический обзор объектно-ориентированных языков программирования. 6](#_Toc43112815)

[1.2. Обзор объектно-ориентированных языков программирования. 8](#_Toc43112816)

[1.3. Плюсы и минусы объектно-ориентированных языков программирования. 9](#_Toc43112817)

[1.4. Основные этапы разработки приложения. 10](#_Toc43112818)

[ВЫВОДЫ К ПЕРВОЙ ГЛАВЕ 11](#_Toc43112819)

[ГЛАВА 2. РАЗРАБОТКА ПРИЛОЖЕНИЯ. 13](#_Toc43112820)

[2.1. Создание окна приложения, класса «Player» и настройка движения. 13](#_Toc43112821)

[2.2. Написание границы игры 16](#_Toc43112822)

[2.3. Создание объектов, которые необходимо ловить. 18](#_Toc43112823)

[2.4. Разработка счетчика, добавление звуков, замена моделей. 22](#_Toc43112824)

[ВЫВОДЫ КО ВТОРОЙ ГЛАВЕ. 25](#_Toc43112825)

[ЗАКЛЮЧЕНИЕ 26](#_Toc43112826)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 27](#_Toc43112827)

[ПРИЛОЖЕНИЕ A. 28](#_Toc43112828)

**АНОТАЦИЯ**

**Близнюк М.А. Разработка аркадной 2D игры**.

В данной работе представляется создание игры на языке Python, разработанного для изучения объектно-ориентированного программирования. Данный курсовой проект может использоваться как инструкция для создания такого же, либо подобного приложения. Программа реализована при помощи языка программирования Python в среде разработки Python-IDLE.

Ключевые слова: Python, объектно-ориентированное программирование (ООП), язык программирования, класс.

**ABSTRACT**

**Bliznyuk M.A. Development of an arcade 2D game.**

This paper presents the creation of a game in Python, designed to study object-oriented programming. This course project can be used as instructions for creating the same or a similar application. The program is implemented using the Python programming language in the Python-IDLE development environment.

Keywords: Python, object-oriented programming (OOP), programming language, class.

**Keywords**: Python, object-oriented programming, programming language, class.

**АНОТАЦІЯ**

**Близнюк М.А. Розробка аркадной 2D гри.**

У даній роботі є створення гри на мові Python, розробленого для вивчення об'єктно-орієнтованого програмування. Даний курсовий проект може використовуватися як інструкція для створення такого ж, або подібного програми. Програма реалізована за допомогою мови програмування Python в середовищі розробки Python-IDLE.

**Ключові слова**: Python, об'єктно-орієнтоване програмування (ООП), мова програмування, клас.

# **ВВЕДЕНИЕ**

**Актуальность темы.** В настоящее время компьютеризация развивается невероятно активно, по сравнению с другими отраслями. Без компьютеров, программ, баз данных не обходится ни одна крупная фирма. Люди все больше проявляют интерес к технологиям, учатся, развлекаются работают с ними.

В современной жизни многие люди увлекаются играми, с нетерпением ждут новые, а также интересуются будущем индустрии. Каждый человек, обладающий возможностью использовать компьютер или телефон, хотя бы однажды встречался с какой-то игрой. Люди могут определять сами, интересующие их жанры, и при помощи различных источников могут выбирать игры из большого, разнообразного списка.

**Цель проекта:** подробно изучить объектно-ориентированное программирование и создать приложение.

В соответствии с целью курсового проекта были поставлены следующие задачи:

* изучить основные принципы объектно-ориентированных языков программирования;
* рассмотреть популярные объектно-ориентированные языки программирования;
* разобрать плюсы и минусы использования ООП;
* привести этапы разработки мобильных приложений;
* спроектировать и разработать игру с использованием выбранного объектно-ориентированного языка программирование и всех его принципов.

**Объектом** курсового проекта является принцип работы программы.

**Предмет** курсового проекта – особенности технологического развития приложений.

**Структура курсового проекта.** Курсовой проект состоит из двух глав, введения, заключения и списка используемой литературы, включает 29 страниц печатного текста, 5 картинок, 27 листингов, 10 источников в списке литературы.

# **ГЛАВА 1. Аналитический обзор объектно-ориентированных языков программирования.**

**1.1 История появления, основные принципы ООП.**

Термин «объектно-ориентированный» зародился в 1960-х годах вместе с языком программирования Simula. Именно тогда были введены понятия объектов и классов, и разработчики то же самое беспокойство, что и сегодняшние разработчики, - как описать сложные системы в простых терминах. Хотя этот язык использовался в различных приложениях, объектно-ориентированное программирование стало набирать популярность лишь после 1980-х годов. С этой популярностью возникла необходимость того, чтобы весь цикл разработки был объектно-ориентированным, и его начали применять в процессе анализа.

Неотъемлемой частью понимания объектно-ориентированного анализа является понятие объекта. Существует множество примеров объектов в повседневной жизни, от книги до автомобиля, от птицы до кошки. Объекты имеют структуру, в которой они обладают собственным поведением, реакцией на различные действия, а также отличия от всех других объектов. Структура определяется переменными, а поведение – методами. Объект обладает возможностью скрывать информацию от других объектов, такой механизм называется **инкапсуляция**. Например, в ситуации, когда покупателю автомобиля необходимо знать лишь цену автомобиля, не обращая внимания на все другие данные помогает инкапсуляция, которая оставляет только всю необходимую информацию. Такая возможность существенно упрощает сопровождение и модификацию ПО.

Группирование объектов в классы – это еще одно важное понятие в понимании объектно-ориентированного языка. Класс может быть определен как прототип, который определяет переменные и методы для всех объектов определенного вида. Классы группируют объекты, которые имеют общие характеристики. Дополнительные объекты могут быть созданы из родительского класса. Эти объекты наследуют общие характеристика класса, а также могут иметь новые. В ООП создание таким механизмом нового класса объекта называется **наследование**. Например, класс транспортного средства может быть определен с характеристиками четырех колес, цвета, дверей, методами движения, переключение передач. Может быть создан новый объект автомашины под названием грузовик, который наследует все характеристики класса автомашины, но к которому можно добавить состояние кузова и другие настройки.

**Полиморфизм** реализовывает возможность реагировать по-разному при получении одной и той же команды. Это позволяет создавать суперкласс объектов, которые имеют общие характеристики, но в которых объекты более низкого уровня реагируют по-разному из-за уникальных характеристик конкретного объекта.

**Абстракция** побуждает сначала думать об объектах в наиболее абстрактной форме, чтобы выделить общие характеристики. Посредством дополнительных итераций процесса абстракции становится возможно установить уникальные характеристики объекта, которые отличают его от всех других объектов.

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Таким образом в данной подглаве было рассказано, когда зародились объектно-ориентированные языки программирования. Выделены основные механизмы, среди которых наследования, инкапсуляция, полиморфизм и абстракция.

# **1.2. Обзор объектно-ориентированных языков программирования.**

Для написания приложений используют множество объектно-ориентированных языков программирования, из которых самыми популярными являются:

1**. C ++** - язык обладает скоростью C с функциональностью классов и объектно-ориентированной парадигмой. Это скомпилированный, быстрый, надежный и мощный язык, используемый во многих приложениях, что он даже используется для создания компиляторов и интерпретаторов для других языков.

2. **Java** – используются во многих сферах, это один из самых популярных и востребованных языков. Девиз Java – «Написано однажды - работает везде», и это отражается на количестве платформ, на которых он работает, и местах, где он используется.

3. **Python** — это высокоуровневый язык программирования общего назначения, который используется в том числе и для разработки веб-приложений. Язык ориентирован на повышение производительности разработчика и читаемости кода. Благодаря большому количество библиотек позволяет написать код в значительно меньшем и читабельном формате.

4. **Ruby** — динамический, рефлективный, интерпретируемый высокоуровневый язык программирования. Язык обладает независимой от операционной системы реализацией многопоточности, сильной динамической типизацией, сборщиком мусора и многими другими возможностями.

5. **C#** — язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML

В курсовой работе был выбран объектно-ориентированный язык Python, так как он обладает всеми необходимыми функциями для создания требуемого приложения.

# **1.3. Плюсы и минусы объектно-ориентированных языков программирования.**

Несмотря на то, что ООП языки очень действенны и развиты, они не полезны для каждой ситуации и поставляются с некоторым багажом, который необходимо учитывать.

**Основные плюсы ООП:**

**Возможность повторного использования** – объектно-ориентированный код является чрезвычайно модульным по своей структуре, из-за полиморфизма и абстракции можно создать одну функцию, которую можно использовать снова и снова, или копировать функциональные возможности, которые уже были написаны с наследованием. Это экономит время, пространство, уменьшает сложность и кодирование.

**Параллельная разработка** – благодаря таким механизмам ООП как абстракция, полиморфизм, наследование и инкапсуляция представляется возможность разрабатывать части программы отдельно друг от друга. Это значительно облегчает параллельную разработку для больших групп разработчиков.

**Обслуживание** – поскольку большая часть кода находится в одном месте, вызывается и используется повторно, этот код гораздо проще поддерживать. Вместо того, чтобы проходить через сотни различных случаев, когда функция вызывается и исправлять каждую по отдельности, мы можем исправить одну модульную и полиморфную функцию, которая вызывается сто раз.

**Безопасность** – в то время как большинство языков имеют некоторую безопасность, объектно-ориентированные языки более удобны, поскольку безопасность, встроенная в инкапсуляцию. Другие методы и классы не могут получить доступ к приватным данным по умолчанию.

**Отражает реальный мир** – поскольку объекты ООП действуют как реальные объекты, код намного проще в использовании и визуализации. Необходимые задачи могут разбираться на реальных примерах из окружающего мира, и поэтому программирование таким способом легче.

**Основные минусы** **ООП:**

**Часто код находится в беспорядке** – поскольку объектно-ориентированные языки хорошо настраиваемы и масштабируемы, может быть легче потерять понимание того, как работает код. ООП код может функционировать по-разному, и существует множество методов для программирования в ООП, которые плохо работают с другими методами, или просто неэффективны или трудны в использовании.

**Требуется больше планирования** – в ООП создание эффективной программы требует четного плана, в большей степени, чем с другими парадигмами программирования. Программисты должны понимать, что они делают, чтобы сделать программу настолько масштабируемой эффективной и мощной, насколько это позволяет объектно-ориентированный язык.

**Непрозрачность** – в объектно-ориентированных языках код разбивается на части и скрывается, из-за чего иногда может быть не понятно, что делает определенный объект или функция. В функциональных же языках весь код располагается на экране.

Таким образом, были разобраны основные плюсы и минусы ООП, определено, что в некоторых ситуациях объектно-ориентированные языки могут быть менее эффективны, чем другие.

# **1.4. Основные этапы разработки приложения.**

Процесс разработки новых приложений состоит из 4-х основных этапов:

1. **Проектирование**. Определяются цели и задачи, способы их решения, а также определяется структура данных и язык программирования, на котором будет написано приложение.

2. **Создание интерфейса**. В программную среду разработки вводятся необходимые управляющие элементы: кнопки, текстовые поля, флажки, переключатели и другие элементы.

3. **Отладка**. Все управляющие элементы связываются программным кодом и путем ввода конкретных значений происходит проверка работоспособности кода и отлавливание возможных ошибок. Логические ошибки самые коварные в этом плане. Этот этап по времени самый длительный.

4. **Заключительный этап**. Идет компиляция кода и создание дистрибутива. Компиляция - процесс перевода программного кода в машинный язык, понятный каждому компьютеру. Здесь же идет подключение необходимых программных библиотек для полной работоспособности приложения. После этого считается, что приложение уже создано и откорректировано.

Итак, в подглаве определены необходимые шаги для создания объектно-ориентированного приложения. Разработка приложения обычно состоит из 4 этапов, среди которых проектирование, создание интерфейса, отладка и заключительный этап.

# **ВЫВОДЫ К ПЕРВОЙ ГЛАВЕ**

Объектно-ориентированные языки появились вместе с языком программирования Simula, и начали активно использоваться после 1980 года. Любой ООП язык использует такие методы как наследование, инкапсуляция и полиморфизм. Данные методы позволяют значительно сократить код благодаря повторному использованию одного и того же класса. Существует множество ООП языков, но самыми популярными является Python, Java и C++.

Были разобраны плюсы и минусы разработки программ на ООП, и определено, что код, выполняющий один и тот же функционал, может быть написан множествами способами, где некоторые из них являются неэффективными и медленными по сравнению с другими. Кроме этого, выделены основные этапы разработки, без которых создание программ невозможно.

# **ГЛАВА 2. РАЗРАБОТКА ПРИЛОЖЕНИЯ.**

# **2.1. Создание окна приложения, класса «Player» и настройка движения**.

Первым делом добавляется библиотека «turtle», при помощи нее будет создаваться персонаж.

Далее при помощи turtle. Screen() получаем экран, в котором будет играть пользователь. Так же настраивается цвет фона и название для экрана.

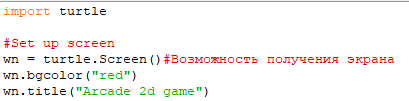


Рис. 2.1. «Листинг»

После при помощи turtle создается класс Player, который наследует от класса Turtle данные, в нем создается конструктор. Далее происходит инициализация библиотеки Turtle и благодаря ей выставляется форма персонажа (треугольник), его цвет (белый) и скорость персонажа.

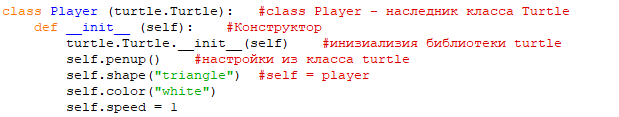


Рис.2.2. «Листинг»

Создаем функцию для движения персонажа и сделаем так, чтобы персонаж мог двигаться вперед со скоростью, которую мы назначили ранее. Затем вызываем класс Player и создадим бесконечный цикл, в котором персонаж будет просто двигаться прямо.

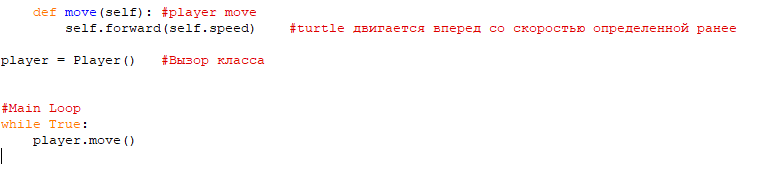


Рис. 2.3. «Листинг»

Общий код выглядит таким образом:

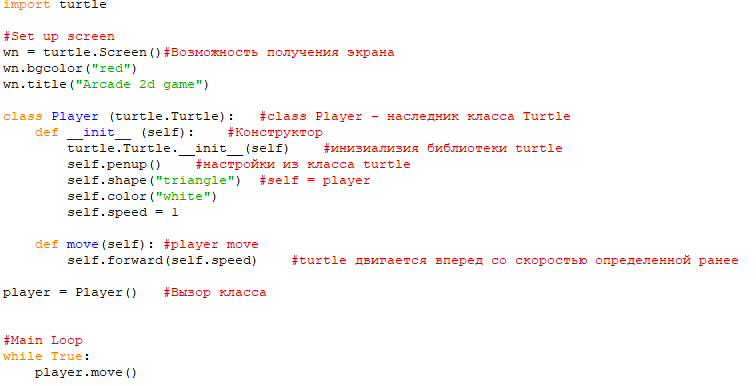


Рисунок 2.4. «Листинг»

При запуске модуля Python выводится окно, в котором треугольник передвигается прямо.

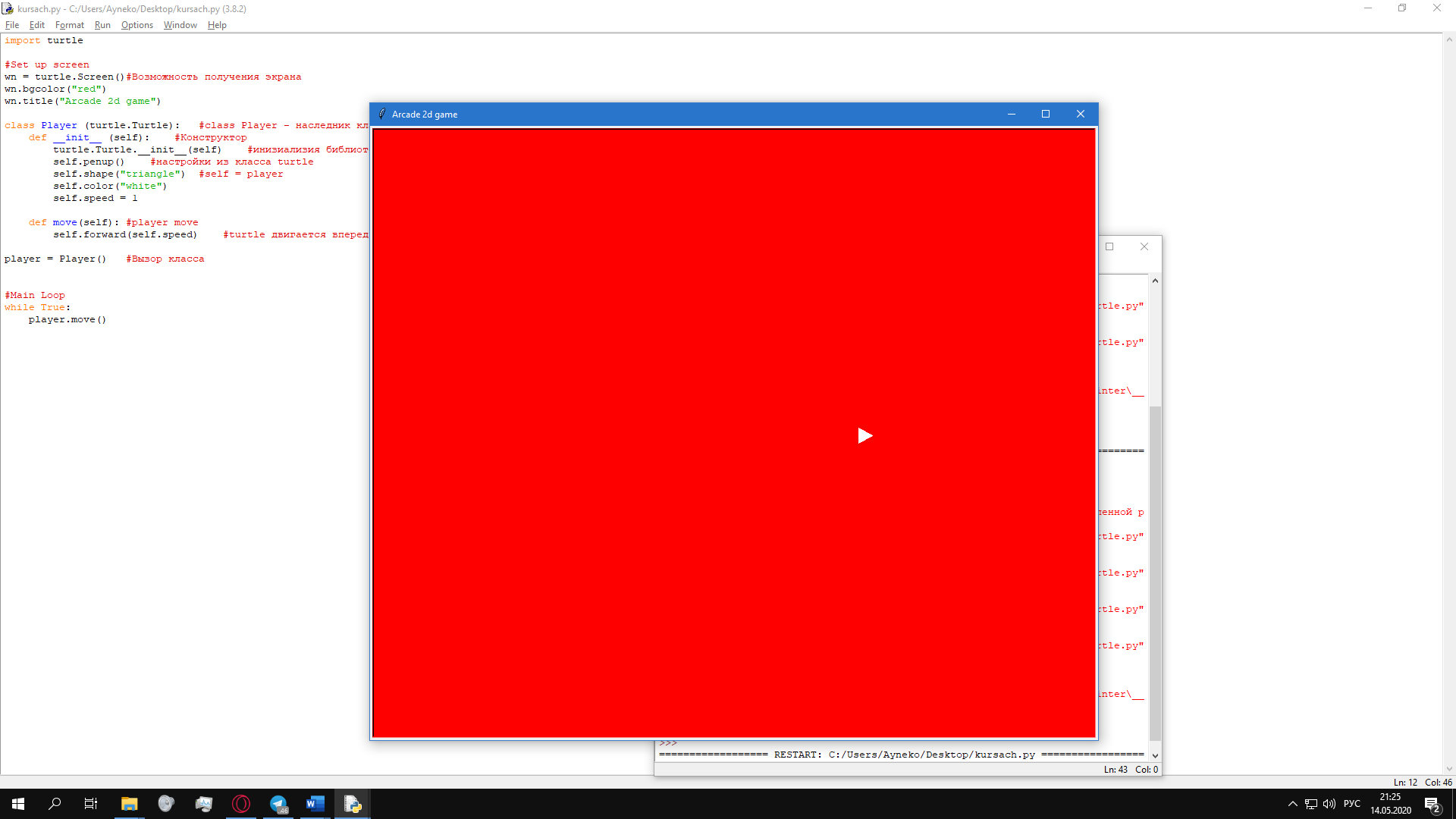
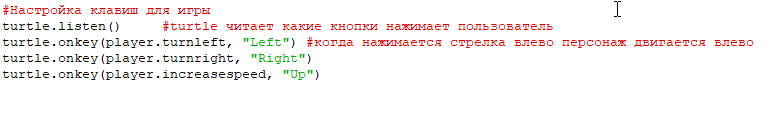
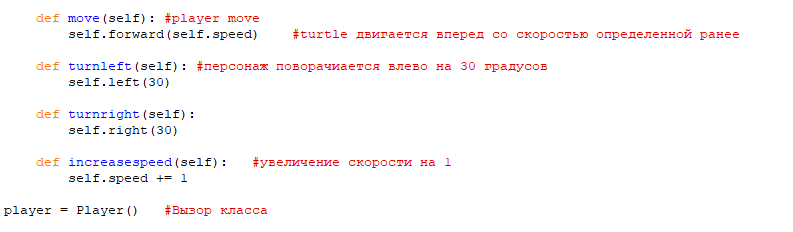


Рисунок 2.1. «Окно игры»

Далее необходимо сделать так, чтобы треугольник мог поворачивать влево и вправо, для этого первым делом при помощи turtle создадим прослушку (turtle.listen), которая будет читать какие кнопки нажимает пользователь, далее на кнопки вправо, влево и вверх создадим вызов функций.(рис. 2.6.):

Рисунок 2.6. «Листинг»

При помощи функций, после нажатия кнопки персонаж будет на 30 градусов поворачиваться в зависимости от того, в какую сторону повернули. Кроме того, при нажатии на стрелочку вверх сделаем возможность повысить скорость персонажа на 1(см. листинг 2.7).

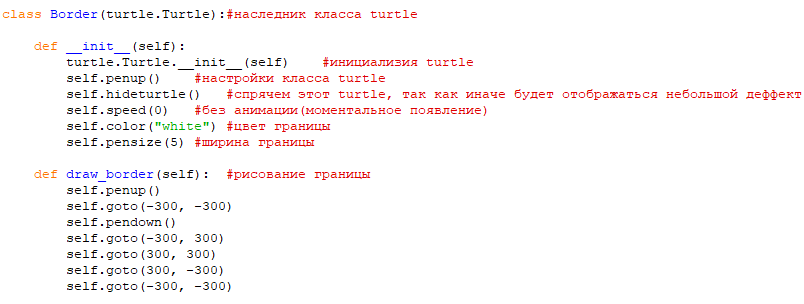


Листинг 2.7

Таким образом был создан класс «player» для которого были написаны функции для перемещения персонажа.

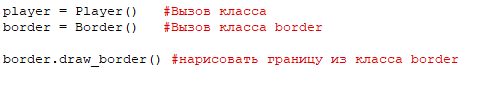
# **2.2. Написание границы игры**

Следующим шагом будет создание границы, для того чтобы треугольник не уходил за область экрана. Для этого создадим класс Border и наследуем от turtle параметры. Произведем инициализацию turtle, спрячем его отображение, так как иначе будет отображаться снизу небольшая стрелка, добавим self.speed(0) который обозначает то, что граница будет рисоваться моментально, без анимации. Так же выставим белый цвет и ширину границы. Далее создадим функцию draw\_border, которая будет рисовать границу. Цифры в self.goto() обозначают количество пикселей, на которые передвигается кисть, pendown() обозначает опустить перо, то есть начать рисовать. (см. листинг 2.8).



Листинг 2.8.

Вызовем класс borderа, а также нарисуем границу (см. листинг 2.9).



Листинг 2.9

При запуске модуля python окно выглядит следующим образом(рис.2.2).

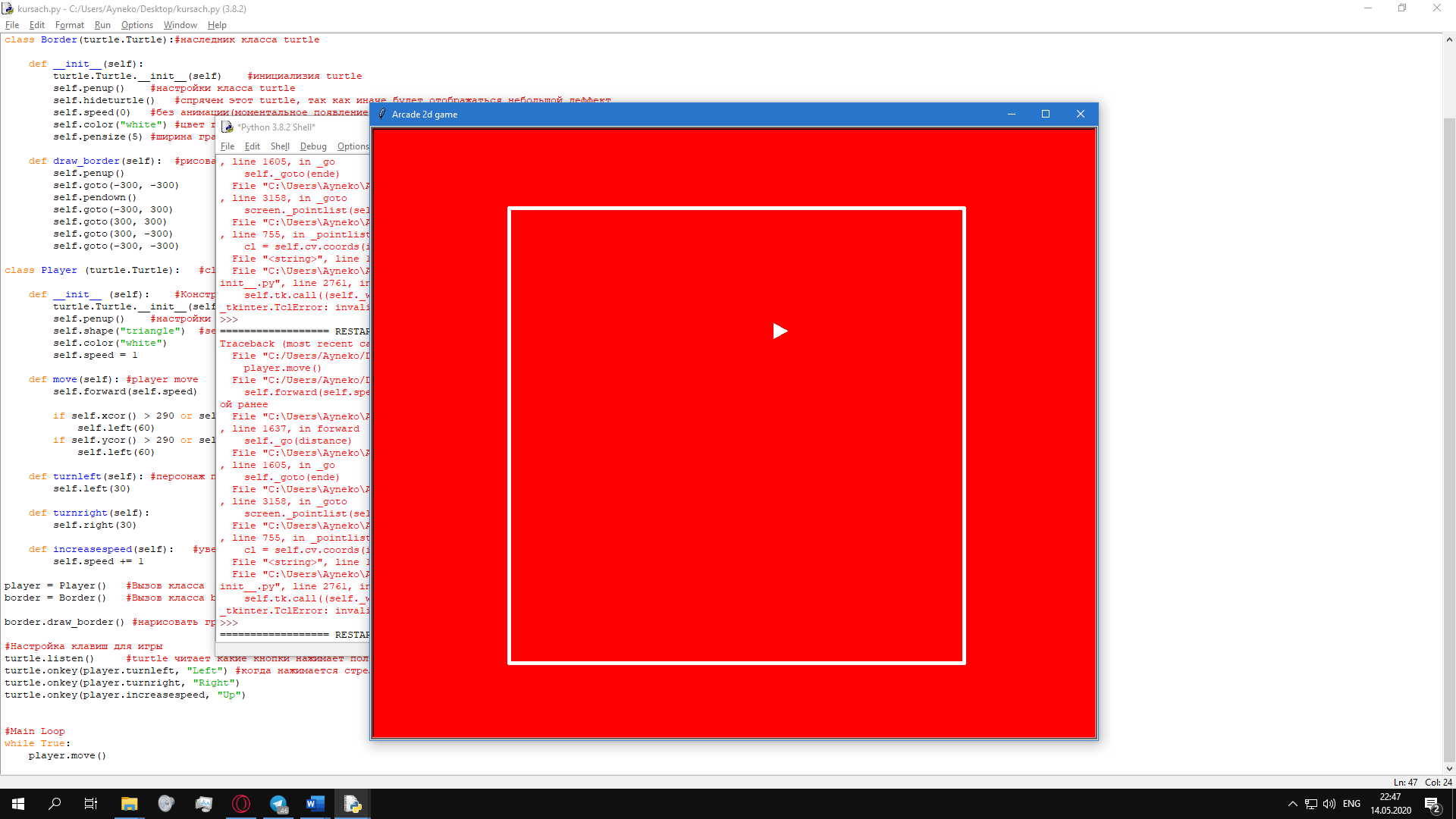
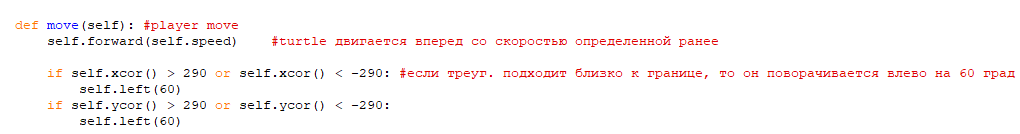


Рис.2.2 «Окно игры»

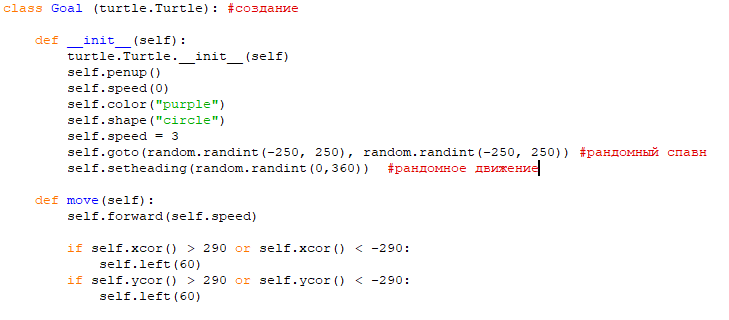
Далее необходимо чтобы персонаж при сталкивании с границей поворачивал в другую сторону для этого в функцию move добавим условия, в которых определим, что при приближении треугольника к границе он осуществит вызов функции поворот влево на 60 градусов (см. листинг 2.10).



Листинг 2.10.

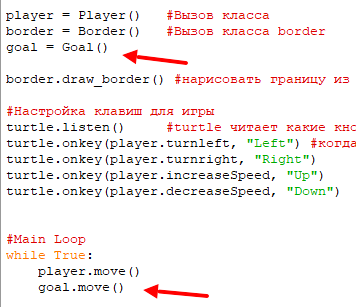
Итак, в подглаве создана граница игры, которая не позволяет классу «player» за нее выйти. При встрече с ней, персонаж будет поворачиваться на 60 градусов и направляться дальше.

# **2.3. Создание объектов, которые необходимо ловить.**

Следующим шагом будет создание объектов, которые персонаж будет ловить для этого по аналогии с классом Player создадим класс Goal, основным его отличием будет случайный появление на поле и случайное движение (см. листинг 2.11). Так же, как и треугольник объект при встрече с границей повернет в другую сторону.

Листинг 2.11

Вызовем класс и в бесконечный цикл добавим для него движение (см. листинг 2.12). Кроме того, по аналогии с ускорением скорости на стрелочку вверх было добавлено замедление движение на стрелочку вниз, до тех пор, пока скорость не будет равна 1.



Листинг 2.12.

После выполнения модуля в python, так же был заменен цвет фона на черный(рис2.3):

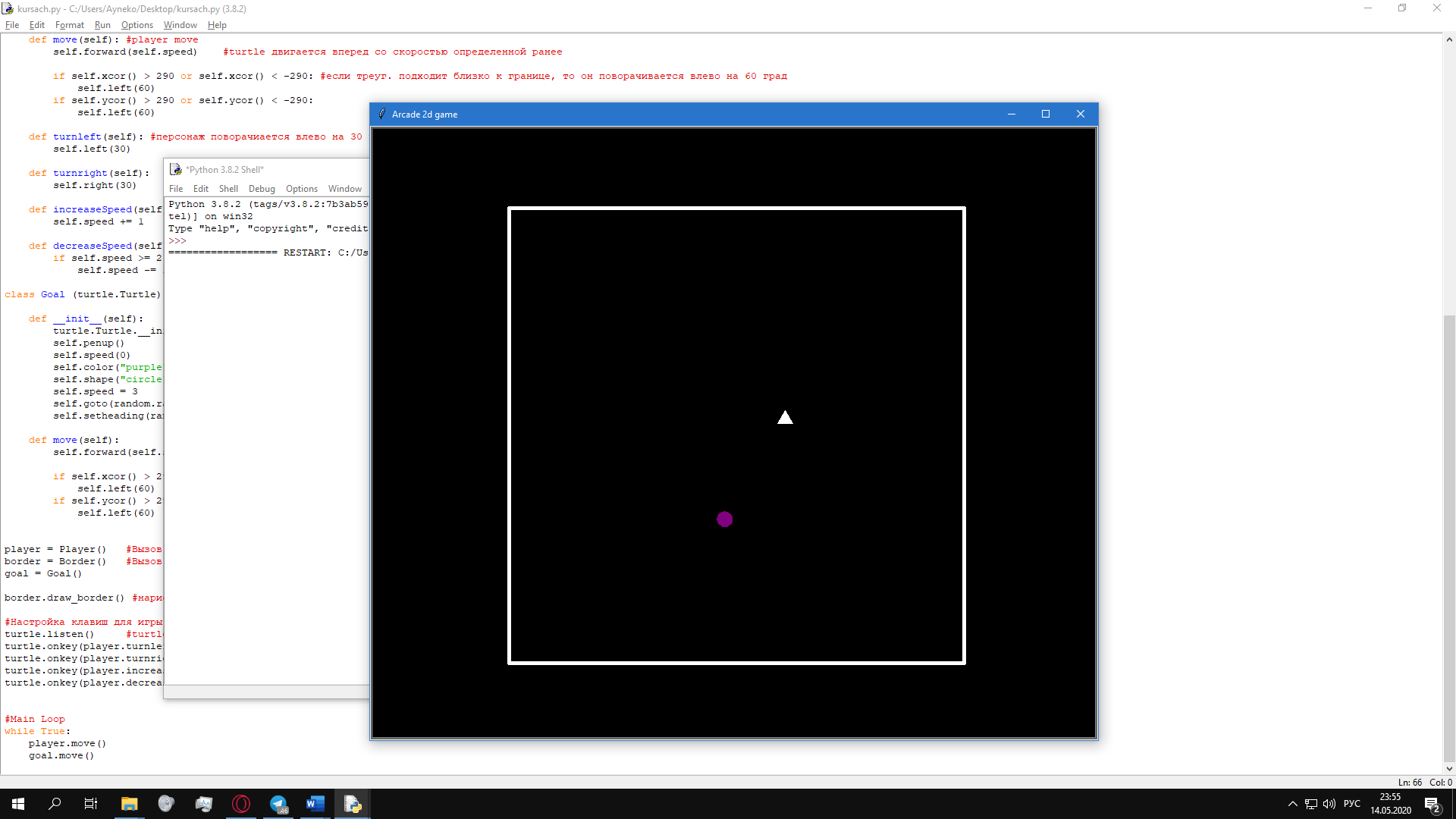
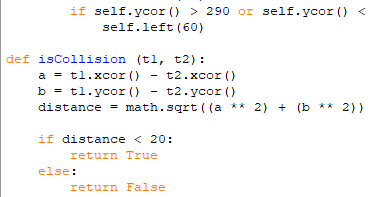


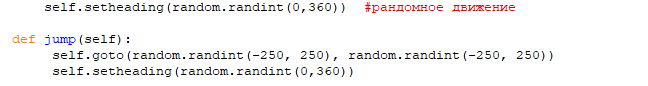
Рис.2.3 «Окно игры»

Затем сделаем так, чтобы при столкновении треугольника с кругом круг пропадал и появлялся снова в случайном месте. Для этого создадим функцию isCollision в которую будут постоянно оправляться параметры Player и Goal, и если дистанция между ними будет маленькой, то вернем True (см. листинг 2.13).



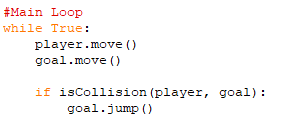
Листинг 2.13

Далее создадим функцию jump, которая вызывается после возращение True в isCollision, она будет в случайном месте размещать новый круг (см. листинг 2.14).



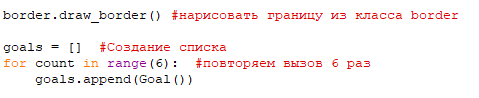
Листинг 2.14.

Так же в основном цикле добавим условие, если функция isCollision возращает True, то выполняется функция jump (см. листинг 2.15).



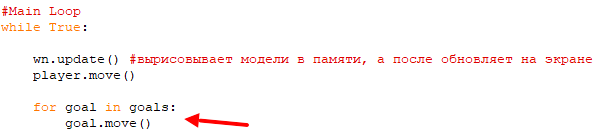
Листинг 2.15.

Теперь увеличим количество кругов, чтобы осуществить это создадим список, в который поместим желаемое количество объектов при помощи цикла for (см. листинг 2.16).



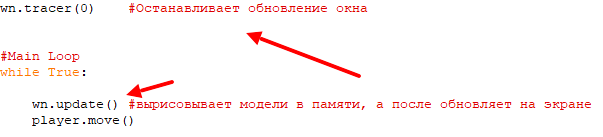
Листинг 2.16.

Далее необходимо в основном цикле добавить цикл, который будет выводить в данном случае 6 кругов и все они будут передвигаться при помощи функции move (см. листинг 2.17).



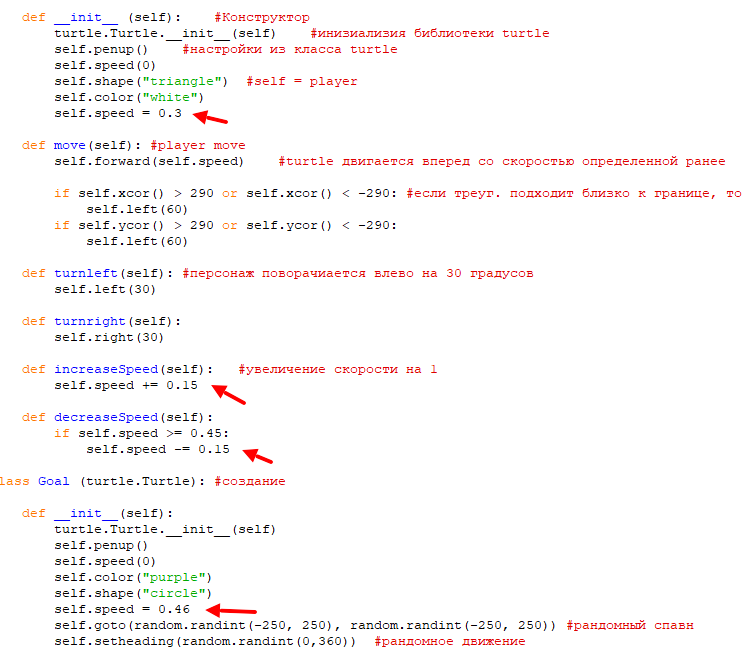
Листинг 2.17

Но теперь появилась проблема, игра стала очень медленной, для того, чтобы это исправить необходимо добавить wn.tracer(), который будет останавливать обновлять каждый раз экран и в основной цикл добавить wn.update(), который вырисовывает модель в памяти, а уже после обновляет на экране. Так же необходимо убрать goal = Goal() из вызова классов, так как теперь он будет вызываться через список (см. листинг 2.18).



Листинг 2.18

После добавления этого кода в зависимости от мощности компьютера игра будет либо нормальной, либо быстрой, чтобы все работало нормально необходимо поменять значения скоростей объектов, в моем случае было изменено скорость Player на 0.3, а скорость goal на 0.46 (см. листинг 2.19)

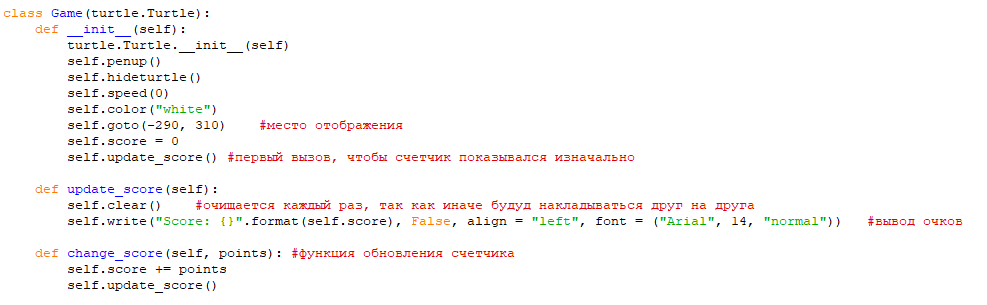


Листинг 2.19

Таким образом был создан класс «goal», который, как и главный персонаж не может выйти за границу поля. Благодаря разработанному списку и вызова цикла на поле будут появляться 6 объектов, когда класс «player» встретится с одним из них круг пропадет и появится в случайной месте.

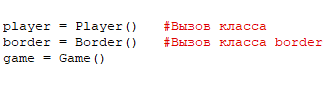
# **2.4. Разработка счетчика, добавление звуков, замена моделей.**

Следующим шагом будет добавление счетчика очков для этого по аналогии с другими создаётся новый класс **Game**, в котором определяется, где он будет находится, значение счетчика изначально назначается 0. Кроме того, создаются функции update\_score и change\_score**. Change\_score** получает параметры self, points и обновляет количество очков, self это класс Game, а points назначаются в основном цикле. **Update\_score** каждый раз при вызове стирает предыдущий вывод на экран, чтобы они не накладывались друг на друга и выводит новые значения (см. листинг 2.20).

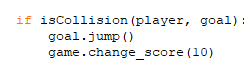


Листинг 2.20

Далее необходимо добавить вызов класса (см. листинг 2.21) и в основной цикл вызвать в функции isCollision изменение очков (см. листинг 2.22).



Листинг 2.21



Листинг 2.22

Таким образом выглядит после запуска кода (рис.2.4):

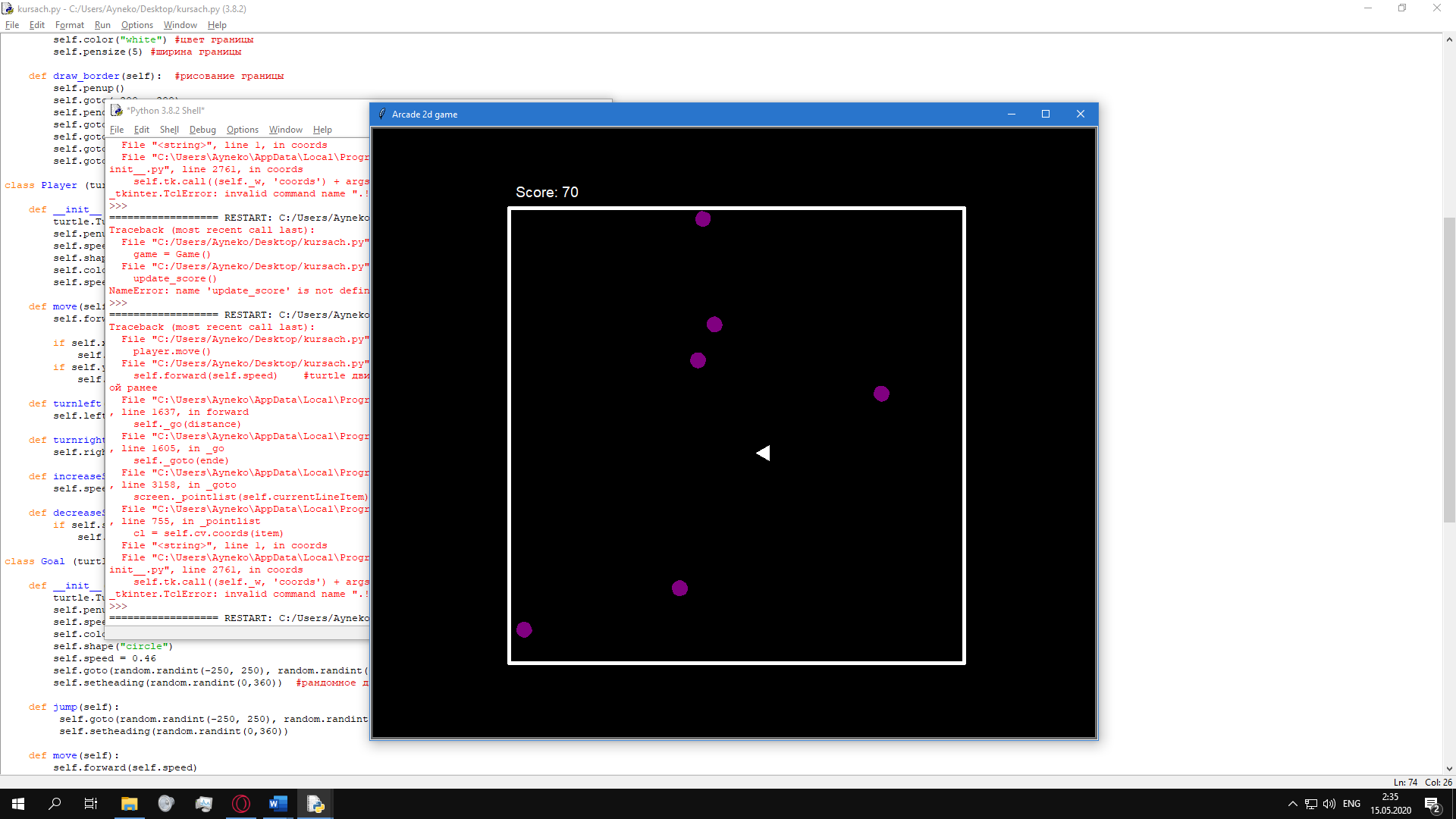
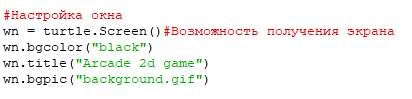


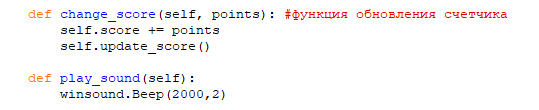
Рис. 2.4. «Окно игры»

Далее изменим фон и добавим звуки, чтобы осуществить это необходимо найти изображение, поместить его в папку с файлом проекта и добавить код wn.bgpic(“название файла”) (см. листинг 2.23)



Листинг 2.23

Чтобы добавить звуки подключим библиотеки «os», «winsound», создадим функцию play\_sound, которая воспроизводит из стандартных звуков windows (работает только на ОС windows). И в isCollision осуществим вызов функции game.play\_sound() (см. листинг 2.24).



Листинг 2.24.

Так же заменим текстуру кругов, для этого зарегистрируем форму в начале кода wn.register\_shape(“название”) и добавим ее в функцию Goal self.shape(“название”).

В данной подглаве были добавлены звуки, заменены модели, фон, а также добавлен счетчик, который увеличивается, когда персонаж встречается с кругом.

# **ВЫВОДЫ КО ВТОРОЙ ГЛАВЕ.**

Во второй главе была разработана игра, в которой были создан главный персонаж, настроено его движение, добавлен фон. Разработаны объекты, которые необходимо ловить персонажем для того, чтобы увеличить показания счета. Кроме того, создана граница игры, за которую не могут выйти другие объекты. Добавлены звуки получения очков, а также установлены модели объектов.

# **ЗАКЛЮЧЕНИЕ**

Объектно-ориентированные языки появились вместе с языком программирования Simula, и начали активно использоваться после 1980 года. Любой ООП язык использует такие методы как наследование, инкапсуляция и полиморфизм.

При рассмотрении объектно-ориентированных языков, были определены самые популярные и выбран язык для написания приложения Python, так как он обладает всеми необходимыми функциями.

Были разобраны плюсы и минусы разработки программ на ООП, и определено, что код, выполняющий один и тот же функционал, может быть написан множествами способами, где некоторые из них являются неэффективными и медленными по сравнению с другими. Благодаря возможности сравнить объекты с окружающим миром ОО языки усваиваются проще, чем другие.

Рассмотрены основные этапы разработки приложения, среди которых выделены проектирование, создание интерфейса, отладка, а также заключающий этап, в котором происходит компиляция кода.

В ходе курсовой работы была создана игра, в которой было создано множество классов, таких как «player», «goal», «border», «game» и т.д., каждый из классов наследуется от отдельной библиотеки и уникален своими характеристиками. Было разобрано как добавить фон, звуки, настроить движения персонажа, сделать так, чтобы объекты не уходили за окно игры.

В будущем для развития игры можно было бы добавить меню, противников, при встрече с которыми игра будет заканчиваться, а также вывод общих и максимальных очков.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Анализ объектно-ориентированных языков [Электронный ресурс].

https://www.umsl.edu/~sauterv/analysis/ooa.html#REF\_1

1. Аналитический обзор языков программирования [Электронный ресурс].

https://studbooks.net/2053084/informatika/postanovka\_zadachi\_obzor\_instrumentov\_realizatsii

1. Объектно-ориентированные языки [Электронный ресурс].

http://bourabai.kz/alg/oop11.htm

1. Плюсы и минусы ОО языков [Электронный ресурс].

https://careerkarma.com/blog/object-oriented-languages/

1. Язык программирования Python [Электронный ресурс].

https://web-creator.ru/articles/python

1. Объектно-ориентированные языки программирования [Электронный ресурс].

https://ru.wikipedia.org/wiki/Объектно-ориентированный\_язык\_программирования

1. Язык программирования С# [Электронный ресурс].

http://progopedia.ru/language/csharp/

1. Этапы разработки приложений [Электронный ресурс].

http://www.programm-school.ru/etapi\_razrabotki.html

1. Этапы разработки приложений [Электронный ресурс].

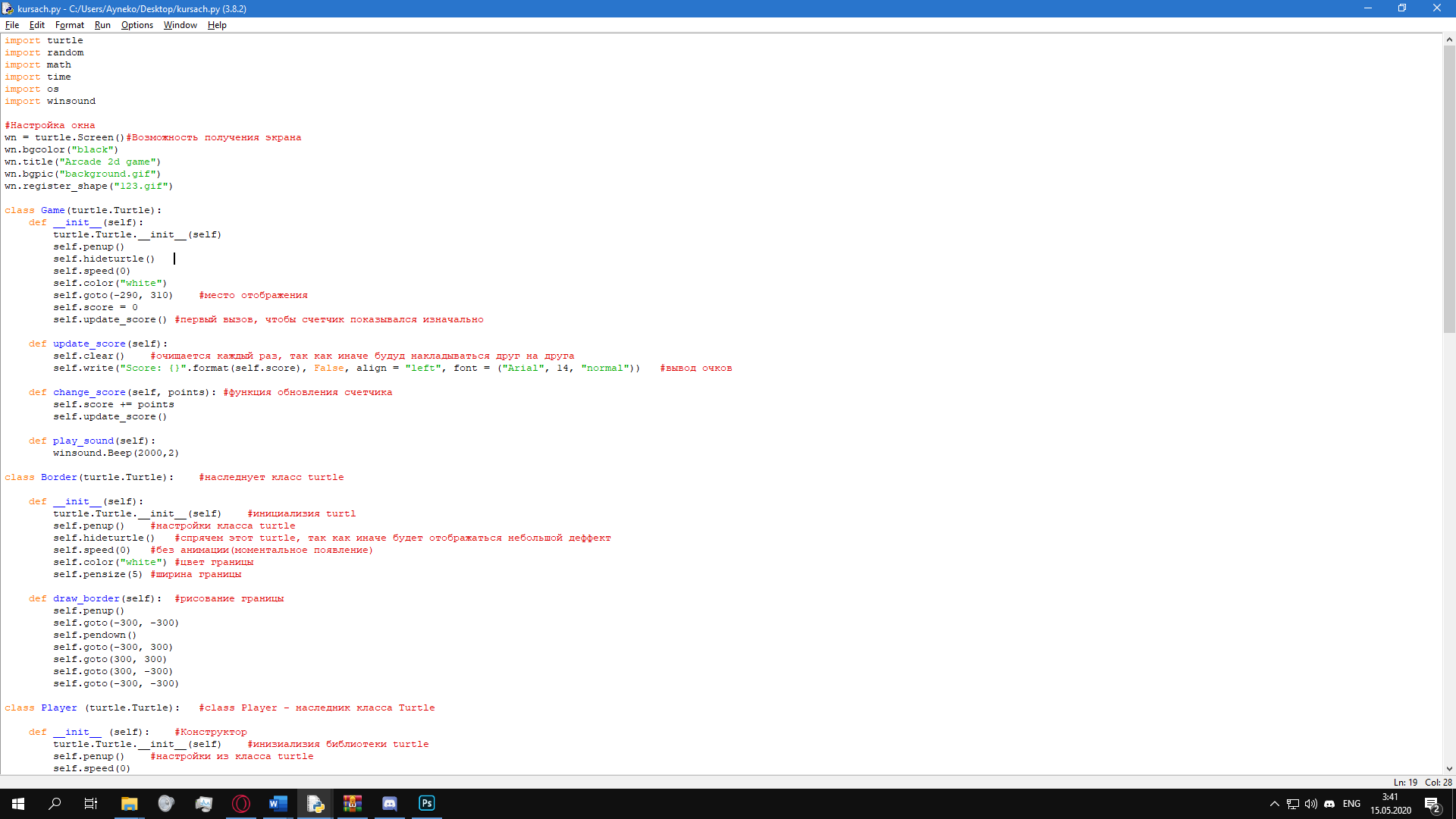
https://woxapp.com/ru/app-development-process/

1. Основные принципы ООП [Электронный ресурс].

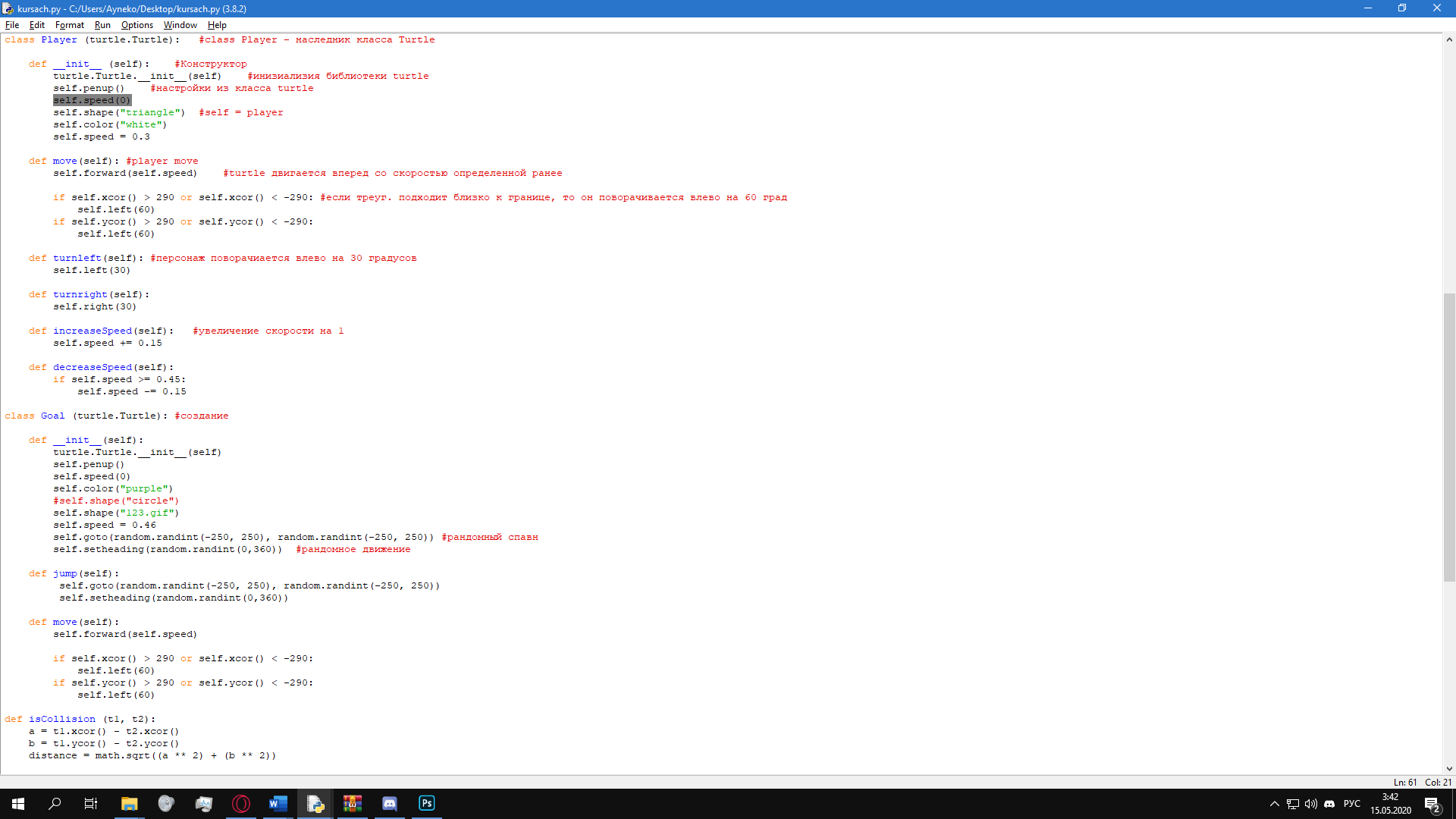
https://labs-org.ru/obektno-orientirovannoe-vizualnoe-programmirovanie/

# **ПРИЛОЖЕНИЕ A.**

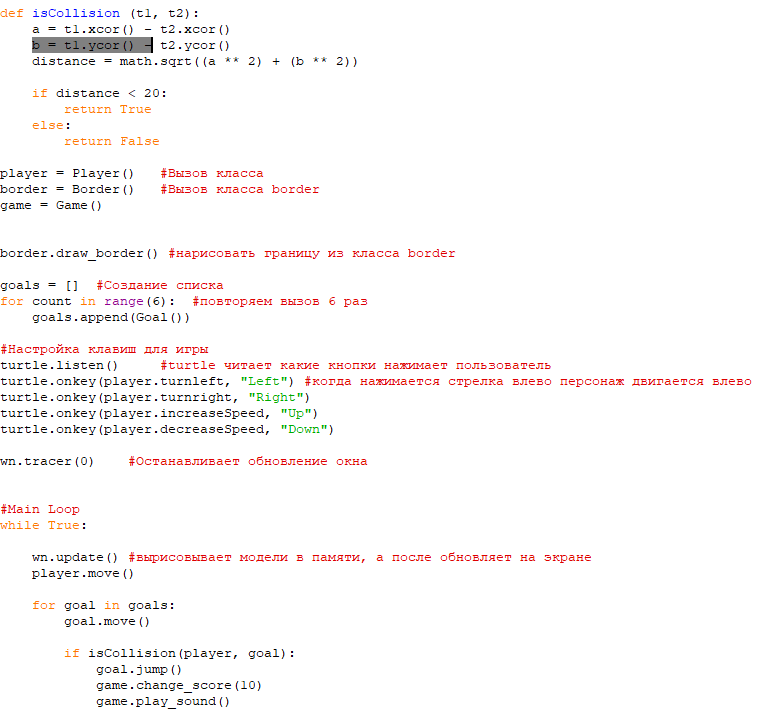
Код программы (см. листинг 3.1-3.3).



Листинг 3.1



Листинг 3.2



Листинг 3.3

Вывод на экран (рис.3.1).

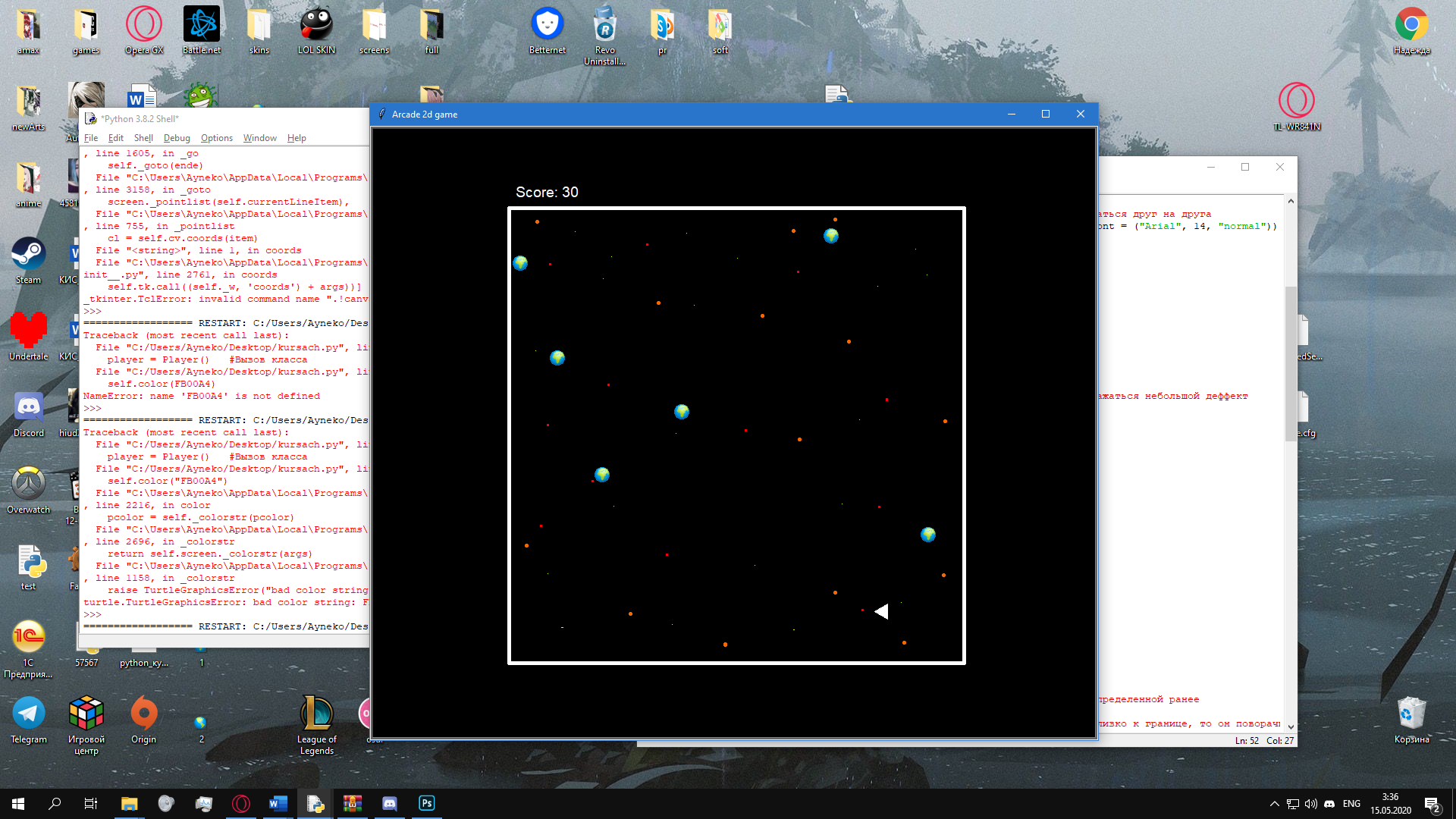


Рис.3.1 «Окно игры»