МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

# КГЭУ «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт ЦТЭ

Кафедра ИИУС

## О Т Ч Е Т

по теме

**«СОЗДАНИЕ ИГРОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ “Арканоид”»**

Соловьёва Леонида Александровича и Матижевой Анны Петровны, обучающихся в группе ПИ-1-22

по образовательной программе направления подготовки

9.03.03 Прикладная информатика  
Направленность: Прикладная информатика в экономике и анализ данных

Прикладная информатика в экономике и анализ данных

ОТЧЕТ ПРОВЕРИЛ

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_/ Шорина Т.В.

Казань, 2023 г.

## СОДЕРЖАНИЕ

Введение .............................................................................................................................. 3

1. Выбор технологии для реализации игрового программного обеспечения………….4
2. Сравнительный обзор инструментов языка программирования Python .................... 7
3. Проектирование алгоритма решения практической игровой задачи программными

средствами языка программирования Python ...................................................................9

* 1. Описание целей и задач .......................................................................................... 9
  2. Описание игрового процесса ............................................................................... ..9

1. Реализация игрового программного обеспечения «Арканоид».

Обоснование его применения в профессиональной деятельности ...............................13

* 1. Основной алгоритм программы ...........................................................................13
  2. Описание классов и функций ...............................................................................14
  3. Обоснование применения игрового программного обеспечения «Арканоид» в профессиональной деятельности .........................................................................14

Заключение ........................................................................................................................ 15

Список использованных источников ...............................................................................16

Приложение 1. Реализация игрового программного обеспечения «Арканоид»…….................................................................................................................17

**ВВЕДЕНИЕ**

Учебная практика (ознакомительная) направлена на закрепление и углубление теоретической подготовки обучающегося, приобретение им практических навыков и компетенций, а также опыта самостоятельной работы в выбранной профессиональной деятельности.

Целью освоение учебной практики является получение первичных профессиональных умений и навыков.

Прохождение учебной практики предполагает выполнение следующих задач:

— закрепление теоретических знаний и умений, приобретенных в предшествующий период теоретического обучения;

— овладение профессиональными навыками решения практических задач;

— развитие первичных профессиональных умений по направлению и профилю подготовки.

## 1. ВЫБОР ТЕХНОЛОГИИ ДЛЯ РЕАЛИЗАЦИИ ИГРОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В качестве технологии для создания игрового ПО был выбран язык программирования Python.

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.

## История

В конце 1989 года Гвидо Ван Россум создал Python – новый интерпретируемый язык программирования, который появился чтобы заменить собой язык ABC, в работе над которым принимал участие автор пайтона. Представленный в 1991 году Python версии 0.9.0 вобрал в себя многие идеи из существовавших на тот момент языков.

Стоит отметить, что Python изначально позиционировался как "язык для каждого", поэтому большой упор делался на читаемость кода и лаконичность синтаксиса.

## Особенности и преимущества языка

— Кроссплатформенность. Python – это интерпретируемый язык, его интерпретаторы существуют для многих платформ. Поэтому с запуском его на любой ОС не должно возникнуть проблем.

— С Python доступно огромное количество сервисов, сред разработки, и фреймворков. Легко можно найти подходящий продукт для работы.

— Возможность подключить библиотеки, написанные на С. Это позволяет повысить эффективность, улучшить быстродействие.

— Наличие самых разных источников информации о Python. Не составит труда найти ответ на любой возникший вопрос, так существует много бесплатной литературы, обучающих видео-пособий, готовых исходников и шаблонов для работы в открытом доступе.

— PEP — единый стандарт для написания кода, что делает код поддерживаемым и читабельным даже при переходе от одного программиста к другому. Это поддерживает популярность Python

— Простота синтаксиса. В Python, из синтаксиса было убрано все лишнее, код чист и понятен, нет операторных скобок, с расставлением которых зачастую возникают сложности.

— Динамическая типизация – это одно из главных достоинств языка Python. Для новичков это возможность упростить написание кода и избежать множества фатальных ошибок и багов в работе.

## Основы синтаксиса

Синтаксис языка Python во многом похож на синтаксис таких языков, как Perl, C и Java, но вместе с этим имеет ряд отличий от этих языков программирования. Одна из первых особенностей Python, которая бросается в глаза программистам, начинающим изучать этот язык программирования, это то, что в нем не используются скобки для обозначения отдельных блоков кода. Вместо них в Python используются двоеточия и отступы.

Количество пробелов в отступах произвольно и выбирается каждым на свое усмотрение, однако по договоренности равняется четырем пробелам. При этом отступ всего блока должен быть одинаковым.

## Структуры данных

Python содержит такие структуры данных как списки (lists), кортежи (tuples) и словари (dictionaries). Списки — похожи на одномерные массивы (но вы можете использовать Список включающий списки — многомерный массив), кортежи — неизменяемые списки, словари — тоже списки, но индексы могут быть любого типа, а не только числовыми. "Массивы" в Python могут содержать данные любого типа, то есть в одном массиве может могут находиться числовые, строковые и другие типы данных. Массивы начинаются с индекса 0, а последний элемент можно получить по индексу -1 Вы можете присваивать переменным функции и использовать их соответственно.

## Функции

Для объявления функции служит ключевое слово «def». Аргументы функции задаются в скобках после названия функции. Можно задавать необязательные аргументы, присваивая им значение по умолчанию. Функции могут возвращать кортежи, в таком случае надо писать возвращаемые значения через запятую. Ключевое слово «lambda» служит для объявления элементарных функций.

## 2. СРАВНИТЕЛЬНЫЙ ОБЗОР ИНСТРУМЕНТОВ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения. Поэтому вместо «присваивания значения переменной» лучше говорить о «связывании значения с некоторым именем». В Python имеются встроенные типы: булевый, строка, Unicode-строка, целое число произвольной точности, число с плавающей запятой, комплексное число и некоторые другие. Из коллекций в Python встроены: список, кортеж (неизменяемый список), словарь, множество и другие. Все значения являются объектами, в том числе функции, методы, модули, классы.

Добавить новый тип можно либо написав класс (class), либо определив новый тип в модуле расширения (например, написанном на языке C). Система классов поддерживает наследование (одиночное и множественное) и метапрограммирование.

Возможно наследование от большинства встроенных типов и типов расширений.

Все объекты делятся на ссылочные и атомарные. К атомарным относятся int, long (в версии 3 любое число int, так как в версии 3 нет ограничения на размер), complex и некоторые другие. При присваивании атомарных объектов копируется их значение, в то время как для ссылочных копируется только указатель на объект, таким образом, обе переменные после присваивания используют одно и то же значение. Ссылочные объекты бывают изменяемые и неизменяемые. Например, строки и кортежи являются неизменяемыми, а списки, словари и многие другие объекты — изменяемыми. Кортеж в Python является, по сути, неизменяемым

списком. Во многих случаях кортежи работают быстрее списков, поэтому если вы не планируете изменять последовательность, то лучше использовать именно их.

Имя (идентификатор) может начинаться с латинской буквы (в Python 3 — буквы любого алфавита в Юникоде, например, кириллицы) любого регистра или подчеркивания, после чего в имени можно использовать и цифры. В качестве имени нельзя использовать ключевые слова и нежелательно переопределять встроенные имена. Имена, начинающиеся с символа подчеркивания, имеют специальное значение.

В каждой точке программы интерпретатор имеет доступ к трем пространствам имен (то есть отображениям имен в объекты): локальному, глобальному и встроенному.

Области видимости имен могут быть вложенными друг в друга (внутри определяемой функции видны имена из окружающего блока кода). На практике с областями видимости и связыванием имен связано несколько правил «хорошего тона», о которых можно подробнее узнать из документации.

Дизайн языка Python построен вокруг объектно-ориентированной модели программирования. Реализация ООП в Python является элегантной, мощной и хорошо продуманной, но вместе с тем достаточно специфической по сравнению с другими объектно-ориентированными языками.

В составе Python поставляется большое число собранных и переносимых функциональных возможностей, известных как стандартная библиотека. Эта библиотека предоставляет массу возможностей, востребованных в прикладных программах, начиная от поиска текста по шаблону и заканчивая сетевыми функциями. Кроме того, Python допускает расширение как за счет ваших собственных библиотек, так и за счет библиотек, созданных сторонними разработчиками. Из числа сторонних разработок можно назвать инструменты создания веб-сайтов, программирование математических вычислений, доступ к последовательному порту, разработку игровых программ и многое другое.

## 3. ПРОЕКТИРОВАНИЕ АЛГОРИТМА РЕШЕНИЯ ПРАКТИЧЕСКОЙ ИГРОВОЙ ЗАДАЧИ ПРОГРАММНЫМИ СРЕДСТВАМИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

### 3.1. Описание целей и задач

На основе вышеизложенного сформулируем цели и задачи проекта.

Целью проекта является: разработка игры «Арканоид» с использованием объектно-ориентированного программирования на языке Python в среде разработки PyCharm Community Edition.

Задачами данного проекта являются:

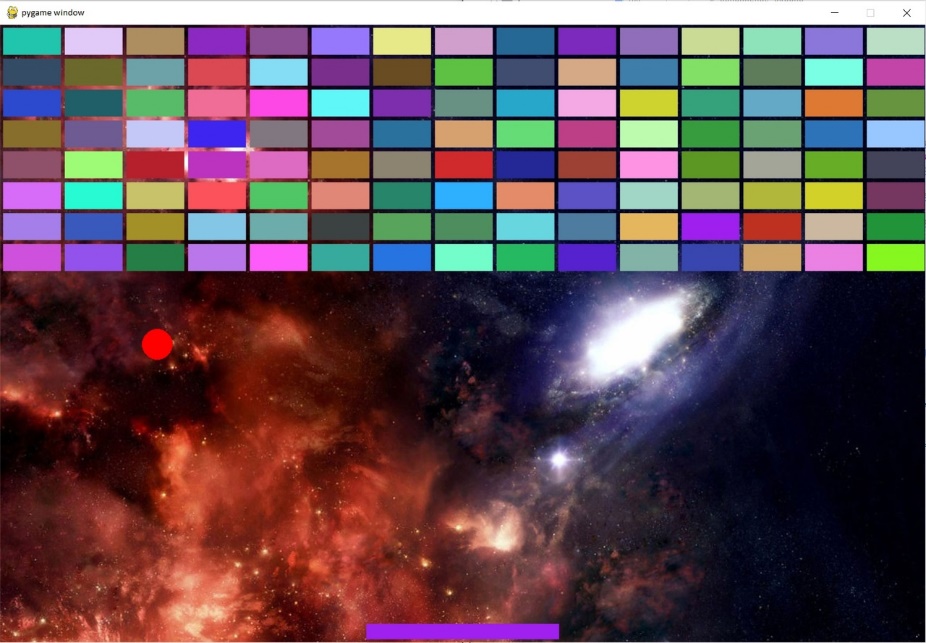
1. Изучение основ проектирования игр.
2. Практика работы с графическими библиотеками.
3. Обучение созданию пользовательского интерфейса.

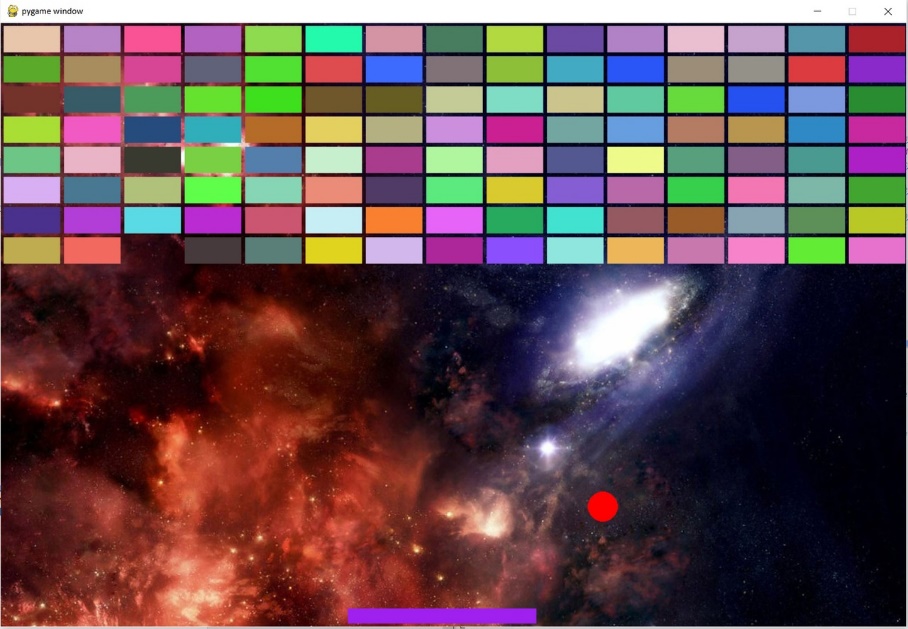
### 3.2. Описание игрового процесса

Игровое поле представляет собой прямоугольное окно размером 1200х800 пикселей. Движущиеся объекты: шарик и платформа. Управляемые объекты: платформа. Суть игры заключается в следующем: игрок должен, не давая шарику упасть, уничтожить все блоки, находящиеся в верхней части окна. Шарик отскакивает от всех границ окна, кроме нижней, так же как и от платформы с блоками. Если мячик упал – игра проиграна, если блоков не осталось – игра выиграна.

Для того, чтобы начать игру необходимо запустить код. Управляющие клавиши: A/D или “Влево”/”Вправо” по желанию. Для того, чтобы выйти из игры, нужно нажать кнопку “Выход ”, находящуюся на окне.

Для наглядного изображения предоставлены скриншоты игры:

  
Рисунок 1. Начальное положение платформы и шарика.

  
Рисунок 2. Шарик и платформа в движении.

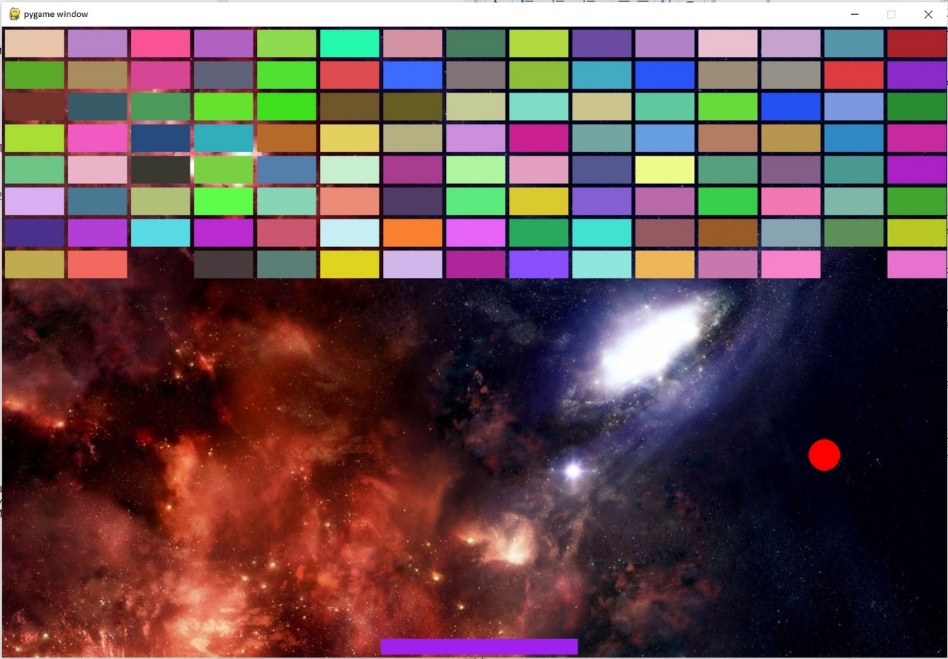
  
Рисунок 3. Игровой процесс.

  
Рисунок 4. Конец игры.

## 4. РЕАЛИЗАЦИЯ ИГРОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «ПОПАДИ В МИШЕНЬ». ОБОСНОВАНИЕ ЕГО ПРИМЕНЕНИЯ В ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

### 4.1. Основной алгоритм программы

На основе предыдущего раздела составим алгоритм работы программы:

1. Подключение библиотек.
2. Создание игрового окна.
3. Описание глобальных констант.
4. Создание изображений игровых объектов.
5. Задать размеры платформе и шарику.
6. Добавить моделям объектов хитбоксы, для реализации столкновений в игре, для этого добавить в классы добавить поле, хранящее невидимый объект, по размеру и форме соответствующий данному экземпляру класса.
7. Для модели платформы реализовать перемещение по горизонтали для возможности отбивать шарик.
8. Для модели шарика реализовать движение, благодаря которому он будет отбиваться от краев игрового поля и платформы, при касании с блоком – удалять его, а так же заканчивать игру, если он коснется нижней границы окна

### 4.2. Описание классов и функций

1. Функция draw\_text() отрисовывает определенный текст определенного цвета и размера шрифта Times New Roman

2. Функция detect\_collision определяет как изменить движение мяча в зависимости от столкновения

**4.3. Обоснование применения игрового программного обеспечения**

## «Арканоид» в профессиональной деятельности

Игра «Арканоид» простая аркадная игра, простая в освоении и не требующая сильного умственного напряжения для прохождения, чтобы скоротать время и скрасить досуг пользователя.

## ЗАКЛЮЧЕНИЕ

В ходе учебной практики был изучен язык Python, разработан алгоритм и программное обеспечение для решения практической игровой задачи «Арканоид» с использованием современных информационных технологий и программных средств языка программирования Python. Задание выполнено в полном объеме. При изложении изученного материала были использованы основные положения правил по содержанию и оформлению технической документации. Для защиты отчета была подготовлен доклад, обосновывающий применение игровой задачи «Арканоид» в профессиональной деятельности.

На основе различных источников были повышены знания и навыки

программирования на языке Python. Получены навыки создания двухмерных игр, изучены необходимые методы для построения игр такого типа. Логика игры реализована в полном объеме: учитывается обработка событий, нажатия клавиш, отрисовка игрового поля и объектов, ведется счет игры. Игра может быть дополнена новыми уровнями, ведением статистики и кастомизацией в зависимости от предпочтений разработчика.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рочев К. В. Информационные технологии. Анализ и проектирование информационных систем. Учебное пособие. СПб.: Лань. – 2019.
2. Филимонова, Е.В. Информационные технологии в профессиональной деятельности: учебник / Филимонова Е.В. – Москва: КноРус, 2019. – 482 с. — ISBN 978-5-406-06532-7.
3. Лутц М. Изучаем Python. Том 1. 5-е изд. Пер. с англ. – СПб.: ООО «Диалектика», 2019 г. – 832 с.
4. Шаньгин В. Ф. Информационная безопасность компьютерных систем и сетей. Учебное пособие для вузов. М.: ИНФРА – М. 2009
5. Васильев Р. Б., Калянов Г. Н., Левочкина Г. А. Управление развитием информационных систем. Учебное пособие. М.: Национальный Открытый

Университет "ИНТУИТ". 2016

**ПРИЛОЖЕНИЕ 1.**

## РЕАЛИЗАЦИЯ ИГРОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «Арканоид»

*# импортируем графическую библиотеку pygame  
import* pygame  
*# импортируем библиотеку random для рандомайзера  
from* random *import* randrange *as* rnd  
*# импортируем библиотеку time для реализации задержки  
from* time *import* sleep  
  
*# инициализация pygame*pygame.*init*()  
  
*# разрешение окна*WIDTH, HEIGHT = 1200, 800  
*# кортеж разрешения для удобства*resolution = (WIDTH, HEIGHT)  
*# количество кадров в секунду*fps = 1  
  
*# создание окна*screen = pygame.display.*set\_mode*(resolution)  
  
*# изображение заднего фона*background\_image = pygame.image.*load*('background.jpg')  
*# позиция для размещения заднего фона*background\_position = (0, 0)  
  
*# ширина платформы*paddle\_width = 250  
*# высота платформы*paddle\_height = 20  
*# скорость движения платформы*paddle\_speed = 3  
*# цвет платформы*color\_of\_paddle = pygame.*Color*('purple')  
*# начальная ширина расположения платформы*start\_width = WIDTH // 2 - paddle\_width // 2  
*# промежуток между экраном и платформой*space\_paddle\_screen = 4  
*# начальная высота расположения платформы*start\_height = HEIGHT - paddle\_height - space\_paddle\_screen  
*# создание платформы*paddle = pygame.*Rect*(start\_width, start\_height, paddle\_width, paddle\_height)  
  
*# радиус мяча*ball\_radius = 20  
*# скорость мяча*ball\_speed = 1  
*# цвет мяча*color\_of\_ball = pygame.*Color*('red')  
*# сторона квадрата, вписанного в окружность мяча*ball\_a = int(ball\_radius \* 2 \*\* 0.5)  
*# создание мяча*ball = pygame.*Rect*(*rnd*(ball\_a, WIDTH - ball\_a), HEIGHT // 2, ball\_a, ball\_a)  
*# переменные, отвечающие за перемещение мяча*dx, dy = 1, -1  
  
*# генерация блоков*block\_list = [pygame.*Rect*(4 + 80 \* i, 4 + 40 \* j, 75, 35) *for* i *in* range(15) *for* j *in* range(8)]  
*# генерация рандомных цветов для блоков*color\_list = [(*rnd*(30, 256), *rnd*(30, 256), *rnd*(30, 256)) *for* i *in* range(15) *for* j *in* range(8)]  
  
  
*# функция для проверки коллизий с блоками  
def* detect\_collision(dx, dy, ball, rect):  
 *# если при столкновении dx > 0  
 if* dx > 0:  
 *# delta x = правая граница мяча - левая граница прямоугольника* delta\_x = ball.right - rect.left  
 *# иначе  
 else*:  
 *# delta x = правая граница прямоугольника - левая граница мяча* delta\_x = rect.right - ball.left  
 *# если при столкновении dy > 0  
 if* dy > 0:  
 *# delta y = нижняя граница мяча - верхняя граница прямоугольника* delta\_y = ball.bottom - rect.top  
 *# иначе  
 else*:  
 *# delta y = нижняя граница прямоугольника - верхняя граница мяча* delta\_y = rect.bottom - ball.top  
  
 *# если небольшая разница между дельтами, то мяч ударился об угол  
 if* abs(delta\_x - delta\_y) < 6:  
 *# полное отражение* dx, dy = -dx, -dy  
 *# если delta x больше чем delta y, то мяч ударился в верхнюю/нижнюю границу  
 elif* delta\_x > delta\_y:  
 *# отражение по вертикали* dy = -dy  
 *# если delta x больше чем delta y, то мяч ударился в вправую/левую границу  
 elif* delta\_y > delta\_x:  
 *# отражение по горизонтали* dx = -dx  
 *# возврат нужных значений  
 return* dx, dy  
  
*# функция для отображения текста  
def* draw\_text(screen, text, x, y, color):  
 *# имя шрифта* font\_name = pygame.font.*match\_font*('Times New Roman')  
 *# создание шрифта с именем и размером 70* font = pygame.font.*Font*(font\_name, 70)  
 *# рендеринг текста* text\_surface = font.*render*(text, *True*, color)  
 *# вычисление рамки текста* text\_rect = text\_surface.*get\_rect*()  
 *# задание расположения для текста* text\_rect.center = (x, y)  
 *# отрисовка текста* screen.*blit*(text\_surface, text\_rect)  
  
*# основной цикл  
while True*:  
 *# ловец событий  
 for* event *in* pygame.event.*get*():  
 *# закрытие окна и остановка программы с помощью "Выход"  
 if* event.type == pygame.QUIT:  
 *# закрытие окна* pygame.display.*quit*()  
 *# остановка программы* exit()  
  
 *# размещение заднего фона в окне* screen.*blit*(background\_image, background\_position)  
 *# отрисовка платформы* pygame.draw.*rect*(screen, color\_of\_paddle, paddle)  
 *# отрисовка мяча* pygame.draw.*circle*(screen, color\_of\_ball, ball.center, ball\_radius)  
 *# отрисовка блоков* [pygame.draw.*rect*(screen, color\_list[color], block) *for* color, block *in* enumerate(block\_list)]  
  
 *# движение мяча по x* ball.x += ball\_speed \* dx  
 *# движение мяча по y* ball.y += ball\_speed \* dy  
  
 *# если ордината центра мяча меньше радиуса мяча, то коллизия с верхней частью поля  
 if* ball.centery < ball\_radius:  
 *# отражаем движение по вертикали* dy = -dy  
  
 *# если абсцисса мяча меньше радиуса или больше ширины окна - радиус, то коллизия с правой и левой частью экрана  
 if* ball.centerx < ball\_radius *or* ball.centerx > WIDTH - ball\_radius:  
 *# отражаем движение по горизонтали* dx = -dx  
  
 *# если мяч столкнулся с платформой  
 if* ball.*colliderect*(paddle):  
 *# отражаем движение по вертикали* dy = -dy  
  
 *# проверка коллизий с блоками* hit\_index = ball.*collidelist*(block\_list)  
 *# если столкновение есть, то hit\_index != -1  
 if* hit\_index != -1:  
 *# удаляем Rect из списка блоков* hit\_rect = block\_list.*pop*(hit\_index)  
 *# удаляем цвет из списка цветов* hit\_color = color\_list.*pop*(hit\_index)  
 *# реализация коллизии* dx, dy = *detect\_collision*(dx, dy, ball, hit\_rect)  
 *# повышение количества кадров в секунду(увеличит скорость игры)* fps += 2  
  
 *# если блоков не осталось  
 if* len(block\_list) == 0:  
 *# отрисовываем текст выигрыша  
 draw\_text*(screen, "YOU WIN!", WIDTH // 2, HEIGHT // 2, pygame.*Color*('white'))  
 *# обновление экрана* pygame.display.*flip*()  
 *# небольшая задержка  
 sleep*(2)  
 *# закрытие окна* pygame.display.*quit*()  
 *# остановка программы* exit()  
  
 *# если ордината центра мяча больше, чем высота окна - радиус, то коллизия с нижней частью поля  
 if* ball.centery > HEIGHT - ball\_radius:  
 *# отрисовываем текст проигрыша  
 draw\_text*(screen, "YOU LOSE!", WIDTH // 2, HEIGHT // 2, pygame.*Color*('white'))  
 *# обновление экрана* pygame.display.*flip*()  
 *# небольшая задержка  
 sleep*(2)  
 *# закрытие окна* pygame.display.*quit*()  
 *# остановка программы* exit()  
  
 *# события нажатых клавиш* keys = pygame.key.*get\_pressed*()  
 *# если нажата кнопка "влево" или "a" и левая граница платформы больше нуля  
 if* (keys[pygame.K\_LEFT] *or* keys[pygame.K\_a]) *and* paddle.left > 0:  
 *# двигаем платформу влево на значение скорости* paddle.left -= paddle\_speed  
 *# если нажата кнопка "вправо" или "d" и правая граница платформы меньше ширины поля  
 elif* (keys[pygame.K\_RIGHT] *or* keys[pygame.K\_d]) *and* paddle.right < WIDTH:  
 *# двигаем платформу вправо на значение скорости* paddle.right += paddle\_speed  
   
 *# обновляем экран* pygame.display.*flip*()