

Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Санкт-Петербургский государственный университет промышленных технологий и дизайна»

**Институт информационных технологий и автоматизации**

Кафедра: Цифровых и аддитивных технологий  
Направление подготовки: 09.03.03 Прикладная информатика  
Профиль подготовки: ИТ-технологии создания цифрового контента

**ОТЧЕТ**

о прохождении учебной практики  
тип практики: технологическая (проектно-технологическая) практика

Руководитель от профильной организации / структурного подразделения СПбГУПТД*: <small>(наименование организации)</small>	     <small>(должность, Ф.И.О., печать организации)</small>	     <small>(подпись, печать)</small>
Руководитель от СПбГУПТД:	Доцент, к.арх., Медведева А.А. <small>(должность, ученая степень / звание, Ф.И.О.)</small>	  <small>(подпись)</small>
Обучающийся:	Соловьёва Екатерина Вячеславовна <small>(Ф.И.О.)</small>	  <small>(подпись)</small>
Курс	2	Учебная группа: 2-МД-20

Санкт-Петербург  
2023

Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное учреждение высшего образования  
«Санкт-Петербургский государственный университет промышленных технологий и дизайна»

**Совместный рабочий график (план) проведения практики и индивидуальное задание**

Вид практики учебная практика

Тип практики технологическая (проектно-технологическая) практика

Обучающийся \_\_\_\_\_ Соловьёва Екатерина Вячеславовна  
(Ф.И.О.)

Институт \_\_\_\_\_ информационных технологий и автоматизации  
(наименование института)

Курс 2 Учебная 2-МД-20 Форма обучения Очная  
\_\_\_\_\_ группа \_\_\_\_\_

Направление подготовки (специальность) 09.03.03 Прикладная информатика  
ИТ-технологии создания цифрового контента

Сроки прохождения практики с 06 февраля 2023г. по 04 июня 2023г.  
(по календарному учебному графику)

Место прохождения практики \_\_\_\_\_ ООО «ЛИСТ»  
(полное наименование организации)

**Совместный рабочий график (план) проведения практики**

Дата	Содержание выполняемых работ и заданий
Общие (типовые вопросы, изучаемые в ходе практики)	
06-08.02.2023	<b>Раздел 1. Исследование предметной области.</b>
	Основные подходы к разработке компьютерных игр.
	Основные принципы моделирование графических объектов.
	Требования к функционированию системы "человек-компьютер". Основные подходы к формированию интерфейса пользователя.
09-11.02.2023	<b>Раздел 2. Компилируемый язык программирования общего назначения.</b>
	История возникновения языка программирования.
	Основы языка программирования.
	Пользовательские функции.
<b>Выполнение индивидуального задания.</b>	
13-04.06.2023	Написание сюжетной линии на основании исследования предметной области.
	Моделирование графических объектов (не только элементы WinForms), либо по средствам консольной графики.
	Формирование таблицы рекордов, хранящихся в файле.

	Разработка алгоритмов программных кодов.
	Написание программных кодов на языке программирования.
	Реализация возможности «выигрывать или проигрывать».
	Включение в игру компьютерных игроков, либо мультиплеера.
	Формирование пояснительной записки (отчет).

**Требования по выполнению и оформлению индивидуального задания** Пояснительная записка оформляется в соответствии с требованиями ГОСТ 7.32-2017 «Отчет о научно-исследовательской работе. Структура и правила оформления» и ГОСТ Р 7.0.100-2018 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления».

**Вид (ы) отчетных материалов по практике и требования к их оформлению  
в соответствии с индивидуальным заданием**

1. *Пояснительная записка*
2. *Презентация по материалам практики для защиты*

Руководитель практики  
от СПбГУПТД \_\_\_\_\_ / Медведева А.А.

Принял к исполнению \_\_\_\_\_ / Соловьёва Е.В.  
(подпись, ф.и.о. обучающегося)

Дата получения обучающимся индивидуального задания 06.02.2023 г.

## РЕФЕРАТ

Пояснительная записка, с. 44, рис. 25, источников 3.

РАЗРАБОТКА ВИДЕОИГР, ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON, СОЗДАНИЕ КОМПЬЮТЕРНОЙ ИГРЫ «SPACE CAT», ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ, ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ, ГРАФИЧЕСКИЕ ОБЪЕКТЫ, ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ

Объектом исследования является высокоуровневый язык программирования Python.

Предмет: создание игры на высокоуровневом языке программирования Python.

Целью исследования является изучение теоретических аспектов создания компьютерных игр, возможностей языка Python и разработка собственной видеоигры.

В ходе работы был проведен анализ предметной области с помощью разнообразной литературы о языке Python и разработке игр с его помощью.

В результате работы была разработана полностью функционирующая аркадная игра под названием «SPACE CAT», написанная на языке программирования Python. Программа была реализована с использованием принципов и методов, изученных в ходе исследования.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1 ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ	9
1.1 Особенности исследования предметной области для разработки компьютерных игр .....	9
1.2 Основные подходы к формированию интерфейса пользователя .....	11
1.3 Основные принципы моделирования графических объектов.....	15
2 ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ.....	17
2.1 Историческая справка о возникновении языка программирования Python.....	17
2.2 Описание функционала языка, которые использовались при разработке компьютерной игры .....	18
2.3 Описание библиотек языка, которые использовались в создании игры .....	19
3 ОПИСАНИЕ ПРОЦЕССА СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ «SPACE SAT» .....	21
3.1 Назначение компьютерной игры.....	21
3.2 Сюжетная линия компьютерной игры.....	22
3.3 Графические объекты и окружение .....	22
3.4 Описание функционала таблицы рекордов .....	26
3.5 Алгоритмы программных кодов.....	27
3.6 Описание функционала программных кодов и представление их скринов .....	29
3.7 Основные принципы оценки результата завершения игры.....	41
3.8 Описание особенностей взаимодействия компьютерных игроков.....	41
ЗАКЛЮЧЕНИЕ .....	42
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	44

## ВВЕДЕНИЕ

Роль языков программирования в создании компьютерных игр является ключевой, поскольку они формируют основу, на которой строятся игры, и воплощают их в жизнь. Языки программирования предоставляют разработчикам необходимые инструменты и синтаксис для проектирования, реализации и оптимизации различных аспектов игры, таких как игровая механика, графика, анимация, звуки. Выбор языка программирования влияет на эффективность, гибкость и общее качество процесса разработки игры.

Несколько языков программирования получили известность в сфере разработки игр благодаря своим возможностям и широкому распространению. Среди самых популярных языков программирования, используемых в разработке игр:

а) C++: он известен своей производительностью, универсальностью и низкоуровневым доступом к оборудованию, что делает его популярным выбором для разработки сложных и ресурсоемких игр. Его способность эффективно управлять памятью и его объектно-ориентированная парадигма делают его подходящим для создания высокопроизводительных игровых движков;

б) C#: язык, разработанный Microsoft, завоевал популярность среди разработчиков игр благодаря простоте использования и интеграции с игровым движком Unity. Он обеспечивает надежную основу для создания игр с богатой графикой, звуком и физическим моделированием;

в) Python: он становится все более популярным в разработке игр благодаря своей простоте, удобочитаемости и обширной экосистеме библиотек и фреймворков. Python часто используется для написания сценариев, создания прототипов игр и разработки ~игровой логики, а критически важные для производительности компоненты реализованы на языках более низкого уровня, таких как C++.

Помимо языков программирования, важно знать о игровых движках. Они предоставляют разработчикам полный набор инструментов и сред для

оптимизации процесса разработки игр. Эти движки выполняют основные задачи, такие как рендеринг графики, управление активами, моделирование физики и обработка ввода. Некоторые из самых популярных игровых движков в отрасли включают в себя:

а) Unity: это широко используемый игровой движок, поддерживающий несколько платформ, включая ПК, консоли, мобильные устройства и виртуальную реальность. Он предоставляет удобный интерфейс, большое сообщество и обширную документацию, что делает его доступным как для начинающих, так и для опытных разработчиков;

б) Unreal Engine: движок, разработанный Epic Games, предлагает мощные инструменты для создания визуально ошеломляющих и очень реалистичных игр. Он известен своими расширенными графическими возможностями, надежной симуляцией физики и широким набором встроенных функций.

Процесс создания игры обычно состоит из нескольких этапов: разработка концепции, дизайн игры, создание ресурсов, программирование, тестирование и развертывание. Гейм-дизайнеры тесно сотрудничают с программистами, художниками и звуковыми дизайнерами, чтобы воплотить свое видение в жизнь. Этап программирования включает в себя реализацию игровой механики, разработку алгоритмов, обработку пользовательского ввода и оптимизацию производительности.

В процессе разработки игровое тестирование имеет решающее значение для улучшения игрового процесса, выявления и исправления ошибок. Инструменты для совместной работы и системы контроля версий часто используются для обеспечения эффективной командной работы и поддержания целостности кода.

Задачи учебной практики направлены на обеспечение всестороннего понимания разработки компьютерных игр с использованием языка программирования Python. Изучая теоретические концепции, анализируя

библиотеки для создания игр и приобретая практические навыки, происходит полноценная подготовка для разработки собственной игры.

В отчете по учебной практике отображено создание игры «SPACE CAT» с помощью библиотеки Pygame языка программирования Python. Игра представляет собой аркаду, в которой необходимо уклоняться от астероидов и стрелять по ним, одновременно с чем собирать монетки, повышая счет, а также усилители, увеличивая скорость.



# **1 ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ**

## **1.1 Особенности исследования предметной области для разработки компьютерных игр**

Изучение предметной области разработки компьютерных игр является важнейшим аспектом проектирования и создания игр. Это включает в себя понимание различных типов компьютерных игр и различных подходов, используемых как крупными студиями, так и независимыми разработчиками. Изучая и анализируя предметную область, разработчики игр могут получить ценную информацию о целевой аудитории, рыночных тенденциях и принципах дизайна, что позволит им создавать успешные игровые проекты.

Компьютерные игры охватывают широкий спектр жанров и стилей игры. Понимание различных типов компьютерных игр имеет решающее значение для разработчиков игр, поскольку это помогает им определить целевую аудиторию и соответствующим образом адаптировать свои проекты. Некоторые распространенные типы компьютерных игр:

а) экшн-игры. Экшн-игры делают акцент на динамичном игровом процессе, часто включающем сражения, исследования и физические испытания. Примеры включают шутеры от первого лица, платформеры и файтинги;

б) ролевые игры (RPG). Ролевые игры включают в себя захватывающее повествование, развитие персонажей и принятие решений. Игроки берут на себя роль главного героя и отправляются на квесты, повышают уровень своих персонажей и участвуют в стратегических сражениях. Примеры включают ролевые игры с открытым миром, пошаговые ролевые игры и массовые многопользовательские ролевые онлайн-игры;

в) стратегические игры. Стратегические игры сосредоточены на принятии тактических решений и управлении ресурсами. Игроки разрабатывают стратегию и планируют свои действия для достижения

конкретных целей. Примеры включают стратегии в реальном времени (RTS), пошаговые стратегии (TBS) и игры в жанре Tower Defense;

е) приключенческие игры. Приключенческие игры делают упор на исследование, решение головоломок и повествовательный опыт. Игроки перемещаются по запутанным средам, взаимодействуют с персонажами и раскрывают историю. Например, игры в стиле «укажи и щелкни», интерактивные игры с рассказыванием историй и графические приключения;

ж) игры-симуляторы. Целью игр-симуляторов является воспроизведение реальных действий или систем. Игроки моделируют различные сценарии, такие как управление городами, управление бизнесом или управление транспортными средствами. Примеры включают симуляторы городского строительства, авиасимуляторы и симуляторы жизни.

Также исследование предметной области разработки игр предполагает изучение различных подходов, используемых крупными студиями и независимыми разработчиками. Под большими студиями понимаются устоявшиеся компании со значительными ресурсами, опытными командами и большими бюджетами. Они часто создают высокобюджетные игры с впечатляющей производительностью и обширными маркетинговыми кампаниями. Примеры крупных студий: Electronic Arts, Ubisoft и Blizzard Entertainment.

С другой стороны, инди-разработчики — это отдельные лица или небольшие группы, которые работают с ограниченными ресурсами и бюджетами. Они часто сосредотачиваются на нишевых жанрах, экспериментальном игровом процессе и уникальных художественных стилях. Инди-разработчики имеют преимущество творческой свободы и гибкости, что позволяет им исследовать инновационные концепции и предлагать более персонализированные и нетрадиционные игровые возможности. Примеры успешных инди-игр включают «Minecraft», «Undertale» и «Celeste».

## **1.2 Основные подходы к формированию интерфейса пользователя**

Пользовательский интерфейс (UI) в компьютерных играх служит важнейшим средством взаимодействия между игроками и игровым миром. Он предоставляет игрокам важную информацию, облегчает игровые действия и улучшает общий игровой процесс. Формирование пользовательского интерфейса требует тщательного рассмотрения принципов дизайна, удобства использования и реализации интерактивных элементов:

1. статические элементы пользовательского интерфейса — это не интерактивные компоненты, которые отображают информацию или предоставляют игрокам визуальные подсказки. Эти элементы остаются фиксированными на экране и обычно используются для передачи важной игровой информации, такой как полосы здоровья, отображение очков или цели миссии. Формирование статических элементов пользовательского интерфейса в Python предполагает использование графических библиотек или фреймворков. Эти библиотеки предоставляют функции для создания и размещения элементов пользовательского интерфейса на игровом экране, что позволяет разработчикам эффективно предоставлять игрокам важную информацию;

2. интерактивные элементы пользовательского интерфейса позволяют игрокам напрямую взаимодействовать с игровым миром или выполнять определенные действия. Эти элементы могут включать кнопки, ползунки, флажки или поля ввода. Формирование интерактивных элементов пользовательского интерфейса в Python осуществляется с использованием парадигмы событийного программирования. Различные библиотеки предоставляют механизмы обработки событий, позволяющие разработчикам обнаруживать вводимые пользователем данные и реагировать соответствующим образом. Например, при нажатии кнопки запускается соответствующее событие, и игровая логика может выполнять необходимые

действия на основе этого события, такие как запуск нового уровня или приостановка игры;

3. HUD (проекционный дисплей). Элементы HUD предоставляют игрокам информацию о состоянии игры в режиме реального времени, сохраняя при этом эффект погружения. Элементы HUD могут включать в себя мини-карты, индикаторы боеприпасов или отображение статуса персонажа. Формирование HUD в Python включает в себя динамическое обновление элементов пользовательского интерфейса в зависимости от текущего состояния игры. Этого можно достичь, постоянно отслеживая и обновляя переменные, которые представляют здоровье игрока, инвентарь или другую соответствующую информацию. Библиотеки языка Python позволяют разработчикам динамически отображать текст или изображения, позволяя создавать элементы HUD, реагирующие на изменения в игровом мире;

4. контекстный интерфейс. Контекстный пользовательский интерфейс относится к элементам пользовательского интерфейса, которые появляются и исчезают в зависимости от контекста игры или конкретных взаимодействий игроков. Эти элементы предоставляют временную информацию, относящуюся к текущей ситуации. Например, когда игрок взаимодействует с объектом, может появиться контекстное меню, предлагающее определенные действия, такие как «использовать», «поднять» или «осмотреть». Формирование контекстного пользовательского интерфейса в Python включает в себя разработку условной логики, которая запускает появление или исчезновение элементов пользовательского интерфейса на основе определенных событий или условий игры.

Формирование пользовательского интерфейса в компьютерных играх требует тщательного рассмотрения принципов проектирования, удобства использования и взаимодействия с игроком. В Python формирование пользовательского интерфейса включает использование графических библиотек или фреймворков, таких как Pygame или Tkinter, для создания статических и интерактивных элементов пользовательского интерфейса.

Кроме того, парадигмы программирования, управляемые событиями, позволяют обнаруживать и обрабатывать вводимые пользователем данные, что позволяет реализовать интерактивные элементы пользовательского интерфейса. Постоянное обновление элементов HUD и создание контекстного пользовательского интерфейса на основе игрового контекста еще больше повышают захватывающий игровой процесс. Применяя эти подходы, разработчики игр могут создавать интуитивно понятные и визуально привлекательные пользовательские интерфейсы, улучшающие игровой процесс и эффективно вовлекающие игроков.

Python предлагает несколько библиотек, которые обычно используются для создания графических интерфейсов для создания игр. Эти библиотеки предоставляют ряд функций для рендеринга графики, обработки пользовательского ввода и управления общим процессом разработки игр. Вот некоторые популярные библиотеки:

1. Pygame — широко используемая библиотека, специально разработанная для разработки игр на Python. Она предоставляет полный набор инструментов и функций для создания 2D-игр. Pygame обрабатывает графику, звук и пользовательский ввод, что делает ее подходящей для разработки как простых, так и сложных игр. Она предлагает такие функции, как обработка спрайтов, обнаружение столкновений и обработка событий. Простота и удобство использования Pygame делают ее популярным среди начинающих разработчиков игр;

2. Pyglet — еще одна мощная библиотека для разработки игр на Python. Основное внимание уделяется созданию визуально привлекательных 2D-игр с аппаратно-ускоренной графикой. Pyglet предоставляет высокоуровневые абстракции для обработки окон, спрайтов, анимации и звука. Она также поддерживает расширенные функции, такие как воспроизведение видео. Универсальность и производительность Pyglet делают ее предпочтительным выбором для разработки игр профессионального уровня;

3. Panda3D — это надежный игровой движок и платформа для разработки, которая поддерживает разработку как 2D-, так и 3D-игр на Python. Она предлагает широкий спектр функций, включая рендеринг, моделирование физики, обработку звука и сетевые возможности. Panda3D предоставляет полный набор инструментов и API для создания захватывающих и визуально красивых игр. Ее гибкость и обширная документация делают ее подходящей как для небольших независимых проектов, так и для крупномасштабного производства;

4. Arcade — это удобная для начинающих библиотека для разработки игр на Python. Она ориентирована на простоту и удобство использования, что делает ее отличным выбором для начинающих программистов. Arcade предоставляет встроенные функции для создания графики, обработки пользовательского ввода и управления игровыми состояниями. Он также предлагает утилиты для обнаружения столкновений, анимации спрайтов и воспроизведения звука. Простой API Arcade и множество обучающего материала делают библиотеку идеальной для изучения концепций разработки игр;

5. Tkinter — стандартная библиотека Python для создания графических пользовательских интерфейсов, включая игровые интерфейсы. Она предоставляет набор компонентов и функций графического интерфейса для создания окон, кнопок, меток и других элементов пользовательского интерфейса. Хотя tkinter не предназначен специально для разработки игр, его можно использовать для создания простых графических интерфейсов для базовых игр или прототипов. Он обычно используется для создания головоломок, настольных игр или текстовых приключений.

Эти библиотеки обеспечивают различные уровни сложности и функций, удовлетворяя различные требования к разработке игр. Выбор библиотеки зависит от таких факторов, как сложность игры, требования к производительности и уровень опыта разработчика. Независимо от выбранной библиотеки универсальность Python и доступные библиотеки для разработки

игр позволяют разработчикам создавать привлекательные и интерактивные графические интерфейсы для своих игр.

### **1.3 Основные принципы моделирования графических объектов**

Игра реализована с помощью библиотеки Pygame. Pygame предоставляет ряд функций и возможностей, которые позволяют разработчикам создавать и управлять графическими объектами в игре. Вот несколько основных принципов, которые следует учитывать при моделировании графических объектов в Python с использованием Pygame:

1. поверхность дисплея. Pygame использует поверхность отображения, на которую выводятся графические объекты. Эта поверхность представляет собой видимую область игрового окна. Он создается с помощью функции `pygame.display.set_mode()` и обычно сохраняется в переменной, например `window` или `screen`. Поверхность дисплея действует как холст для рисования и обновления графических объектов;

2. система координат. Pygame использует систему координат, в которой начало координат (0, 0) находится в верхнем левом углу поверхности дисплея. Координата `x` увеличивается при движении вправо, а координата `y` увеличивается при движении вниз. Важно понимать, как работать с этой системой координат при позиционировании и перемещении графических объектов;

3. спрайты. Спрайты являются основными строительными блоками графических объектов в Pygame. Спрайт представляет собой изображение или визуальный элемент в игре. Pygame предоставляет класс `pygame.sprite.Sprite` для создания спрайтов и управления ими. Спрайты можно загружать из файлов изображений с помощью таких функций, как `pygame.image.load()`, или создавать программно. Спрайты могут иметь такие свойства, как положение, размер и представление изображения;

4. рисование и визуализация. Pygame предоставляет функции для рисования и рендеринга графических объектов на поверхности дисплея. Модуль `pygame.draw` предлагает различные функции для рисования основных фигур, таких как прямоугольники, круги, линии и многоугольники. Кроме того, спрайты могут отображаться на поверхности экрана с помощью класса `pygame.sprite.Group` и его метода `draw()`. Это позволяет эффективно отображать и обновлять несколько спрайтов одновременно;

5. анимация. Анимация графических объектов включает изменение их свойств, таких как положение или внешний вид, с течением времени для создания движения или визуальных эффектов. Pygame поддерживает анимацию, предоставляя функции для обновления положения, изображения или других атрибутов спрайтов. Многократно обновляя эти свойства в игровом цикле, можно добиться плавных эффектов анимации;

6. обнаружение столкновений. Обнаружение столкновений является важным аспектом моделирования графических объектов в играх. Pygame предоставляет встроенные функции обнаружения столкновений, такие как `pygame.sprite.spritecollide()`, для проверки столкновений между спрайтами или другими игровыми объектами. Эти функции позволяют разработчикам определять поведение при столкновении и реализовывать игровую механику на основе взаимодействия объектов.



## **2 ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ**

### **2.1 Историческая справка о возникновении языка программирования Python**

Язык программирования Python был задуман и разработан Гвидо ван Россумом в конце 1980-х – начале 1990-х годов. Гвидо ван Россум, голландский программист, хотел создать язык программирования, отличающийся простотой, удобочитаемостью и простотой использования. Он стремился разработать язык, который был бы одновременно мощным и доступным для программистов всех уровней опыта.

Разработка Python началась в декабре 1989 года, когда Гвидо ван Россум начал работу над преемником языка программирования ABC. Гвидо назвал свой новый язык «Python» в честь британской комедийной группы Monty Python, так как он был поклонником их юмора. Выбор имени отражал его желание сделать язык веселым и доступным.

Ранняя разработка Python была сосредоточена на создании языка с чистым и удобочитаемым синтаксисом. Гвидо стремился устранить ненужные сложности и снизить когнитивную нагрузку на программистов. В Python принят синтаксис со значительным количеством пробелов, в котором отступы играют решающую роль в структуре кода, повышая читабельность кода.

Первый общедоступный выпуск Python версии 0.9.0 состоялся в феврале 1991 года. Он привлек внимание и быстро привлек небольшую, но преданную базу пользователей. Гвидо ван Россум продолжал руководить разработкой Python, улучшая язык и добавляя новые функции.

С момента своего появления в 1991 году Python стал одним из самых популярных динамических языков программирования наряду с Perl, Ruby и другими [3, с. 14]. Его простота, универсальность и обширная стандартная библиотека сделали его привлекательным выбором для различных приложений. В 2000 году был выпущен Python 2.0, в котором были внесены

значительные улучшения и новые функции, но при этом была сохранена обратная совместимость с более ранними версиями.

Python 3.0, основная версия языка, была выпущена в декабре 2008 года. Она включала несколько изменений и улучшений, направленных на устранение некоторых ограничений и несоответствий Python 2.x.

С годами популярность Python продолжала расти. Python стал популярным языком для веб-разработки, анализа данных, машинного обучения, искусственного интеллекта и автоматизации, а также для разработки игр.

## **2.2 Описание функционала языка, который использовался при разработке компьютерной игры**

При разработке компьютерной игры был использован следующий функционал языка:

1) `import` — инструкция, используемая для подключения модулей и библиотек;

2) `open()` — открывает файл для чтения или записи файлового потока;

3) `len()` — возвращает число элементов в указанном объекте.

Также использованы циклы:

1) `for` — цикл, предназначен для перебора элементов структур данных и других составных объектов;

2) `while` — цикл “пока” позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно.

Условные операторы:

1) `if` — оператор ветвления, позволяет выполнить определенный набор инструкций в зависимости от некоторого условия;

2) `else` — предусматривает альтернативный заданному в `if` условию вариант, когда при истинном условии нужно выполнить один набор инструкций, при ложном — другой;

3) `elif` – используется для реализации выбора из нескольких альтернатив.

Также в работе использовано объектно-ориентированное программирование. Объектно-ориентированное программирование (ООП) — это подход, при котором программа рассматривается как набор объектов, взаимодействующих друг с другом. У каждого есть свойства и поведение. ООП ускоряет написание кода и делает его более читаемым. Использовано наследование – пользовательские классы спрайтов наследуются от встроенного в библиотеку Pygame класса `Sprite`.

### **2.3 Описание библиотек языка, которые использовались в создании игры**

В работе были использованы библиотеки Pygame, Random, CSV. Они подключаются в программу в качестве дополнительных модулей. Модуль — это файл, содержащий определения и операторы Python. Самый простой способ импортировать модуль в Python – это воспользоваться конструкцией: `import имя_модуля` [2, с. 73].

Pygame — популярная библиотека для создания видеоигр и мультимедийных приложений на Python. Она предоставляет функциональные возможности для обработки графики, звука и пользовательского ввода. Pygame построена на основе библиотеки Simple DirectMedia Layer (SDL), которая обеспечивает межплатформенную поддержку. Она упрощает разработку игр, предоставляя инструменты и модули для таких задач, как создание игровых окон, обработка спрайтов, управление столкновениями, воспроизведение звуков и многое другое. С помощью Pygame разработчики могут сосредоточиться на игровой логике и дизайне, используя встроенные функции библиотеки для разработки игр.

Модуль Random — это стандартная библиотека Python, предоставляющая функции для генерации случайных чисел и выбора случайных элементов из последовательностей. Модуль генерирует

псевдослучайные числа для нескольких различных распределений [1, с. 36]. Модуль Random часто используется в различных приложениях, включая игры. Внедряя случайность в программы, разработчики могут создавать непредсказуемое поведение, внедрять алгоритмы рандомизации или моделировать реальные сценарии, в которых требуется случайность.

CSV — это формат файла, обычно используемый для хранения табличных данных, таких как электронные таблицы или базы данных. Это обычный текстовый формат, в котором каждая строка представляет строку данных, а значения в строке разделяются запятыми (или другими разделителями). Формат CSV широко поддерживается программным обеспечением для работы с электронными таблицами и базами данных, что упрощает импорт и экспорт данных. В Python модуль CSV является частью стандартной библиотеки и предоставляет функции для чтения и записи в файлы CSV. Это упрощает такие задачи, как анализ данных CSV, доступ к отдельным ячейкам или столбцам и манипулирование данными для дальнейшей обработки или анализа.

### **3 ОПИСАНИЕ ПРОЦЕССА СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ «SPACE CAT»**

#### **3.1 Назначение компьютерной игры**

Назначение компьютерной игры под названием «SPACE CAT» состоит в том, чтобы сделать игровой процесс интересным и увлекательным. Игра вращается вокруг управляемого игроком персонажа-кошки в космической среде. Цель состоит в том, чтобы управлять кошкой и перемещаться по игровому миру, избегая препятствий и собирая очки.

Игра предлагает несколько функций и механик для улучшения игрового процесса. Игрок может перемещать персонажа-кошку с помощью клавиатуры, что позволяет ему свободно исследовать игровой мир. Кошка может двигаться влево, вправо, вверх и вниз, чтобы избежать приближающихся астероидов и собирать монеты. Счет игрока увеличивается с каждой собранной монетой, вознаграждая умелую навигацию и точные движения.

Кроме того, в игре представлены различные игровые элементы, чтобы добавить вызов и азарт. Игрок должен уворачиваться от астероидов, спускающихся с верхней части экрана, так как столкновение с ними приводит к окончанию игры. Иногда появляются предметы усиления, которые при сборе увеличивают скорость кота. Это добавляет элемент вознаграждения за риск, поскольку игрок должен взвесить преимущества сбора бонусов и возросшую сложность управления более быстрой кошкой. В игре также есть пулевые снаряды, которыми игрок может стрелять, чтобы уничтожать астероиды, добавляя элемент наступательного игрового процесса.

В целом, цель этой компьютерной игры состоит в том, чтобы обеспечить развлечение, бросить вызов навыкам и рефлексам игроков. Она сочетает в себе элементы игрового процесса, основанного на навыках, уклонения от препятствий и механики сбора, чтобы создать приятный и захватывающий игровой процесс.

### **3.2 Сюжетная линия компьютерной игры**

Сюжетная линия сосредоточена вокруг бесстрашного кота, который берет на себя роль космического авантюриста. Действие игры происходит в альтернативной реальности, где коты защищают Землю от падения астероидов.

На первом уровне игры игрок управляет космическим котом, который должен перемещаться в космосе и избегать столкновений с астероидами. Цель игрока – собирать монеты и бонусы, уклоняясь от препятствий. Чем больше монет собирает игрок, тем выше его счет. На втором уровне игрок может стрелять пулями-хомячками по астероидам, чтобы уничтожить их и заработать дополнительные очки.

Если игроку не удастся увернуться от астероида или он сталкивается с ним, появляется анимация взрыва, указывающая на конец игры. В этот момент игроку предоставляется экран завершения игры, на котором отображается его окончательный счет и варианты перезапуска игры или просмотра таблицы рекордов.

### **3.3 Графические объекты и окружение**

Для разработки графических объектов была использована программа Adobe Photoshop, в которой с помощью инструмента «выделение» были вырезаны нужные спрайты.

Кот является главным и единственным действующим лицом игры, он изображен на рисунке 1.



Рисунок 1 – Изображение кота

Враги в игре представлены астероидами, их изображение представлено на рисунке 2.



Рисунок 2 – Изображение астероида

Спрайт для повышения игрового счета (монеты) и усиления (подорожник) представлены на рисунках 3 и 4 соответственно.



Рисунок 3 – Изображение монеты

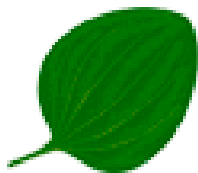


Рисунок 4 – Изображение усиления

Спрайт взрыва, появляющийся при столкновении с астероидом, представлен на рисунке 5.

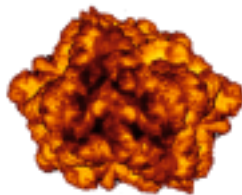


Рисунок 5 – Изображение взрыва

Пуля, которой можно выстреливать по астероидам, представлена на рисунке 6.



Рисунок 6 – Изображение пули

Задний фон игры – локация – представлен на рисунке 7.



Рисунок 3 – Локация

На рисунке 8 представлено то, как выглядит запущенная игра. Сверху посередине отображен текущий счет и уровень.



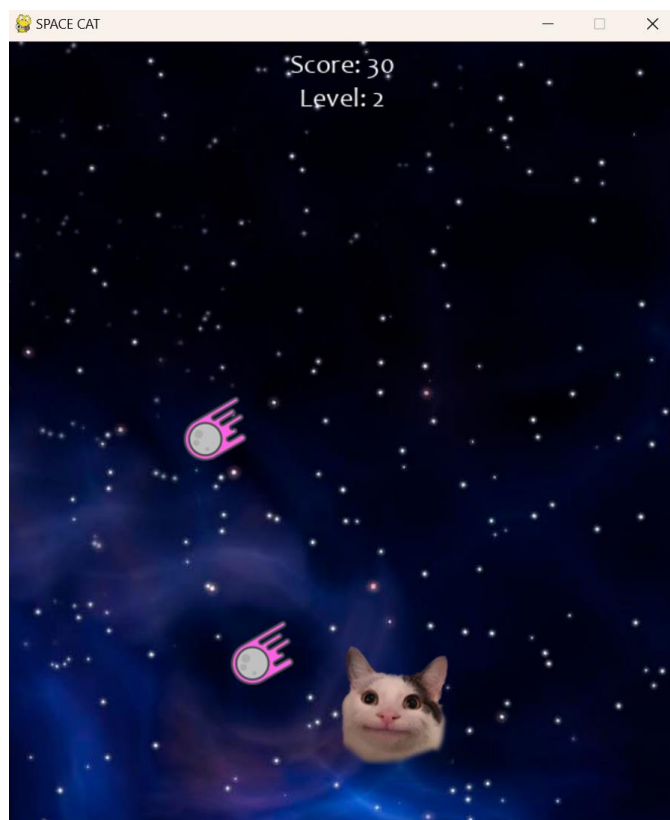


Рисунок 8 – Процесс игры

На рисунке 9 представлен экран проигрыша. Отображена надпись, сообщающая об окончании игры, счет, и две кнопки: перезапуск и рекорды.

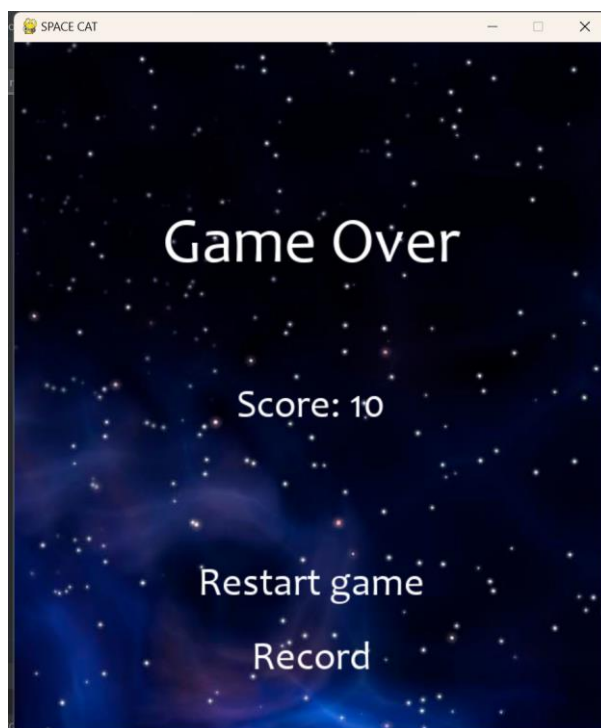


Рисунок 9 – Экран проигрыша

На рисунке 9 представлен экран выигрыша. Отображена надпись, сообщающая о победе.



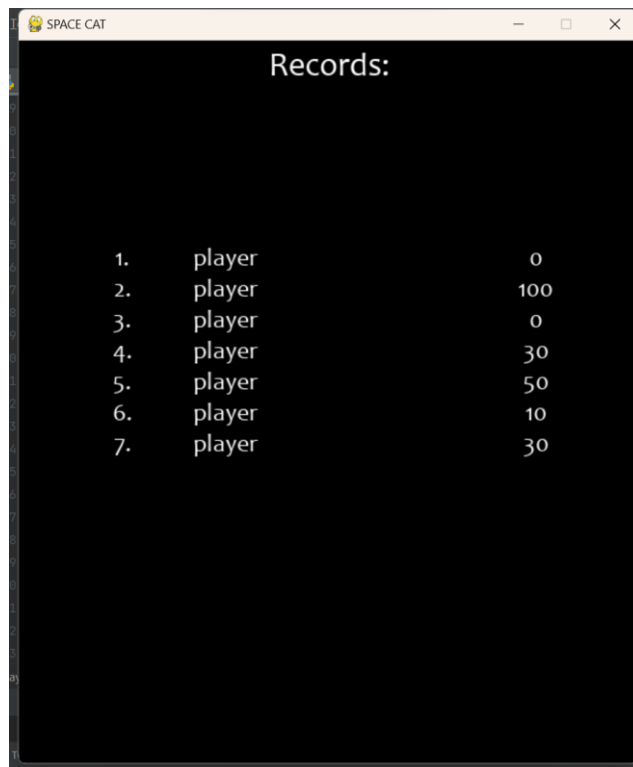
Рисунок 10 – Экран выигрыша

### **3.4 Описание функционала таблицы рекордов**

Таблица рекордов — это отображение баллов, достигнутых игроками. Таблица реализована с использованием формата файла CSV, что позволяет игре легко читать файл и делать новые записи.

Когда игрок обращается к таблице рекордов, игра считывает содержимое файла «records.csv». Каждая запись в файле состоит из имени игрока и его соответствующего счета. Если количество записей превышает 10 штук, файл очищается.

Таблица рекордов, отображаемая в игре, представлена на рисунке 11.



1.	player	0
2.	player	100
3.	player	0
4.	player	30
5.	player	50
6.	player	10
7.	player	30

Рисунок 11 – Таблица рекордов

### 3.5 Алгоритмы программных кодов

Для описания алгоритма существуют несколько способов:

1. естественный язык;
2. псевдокод;
3. графический способ с помощью блок-схемы или диаграммы;
4. код на языке программирования.

Был выбран графический способ представления в виде блок-схемы. Она представлена на рисунке 12.

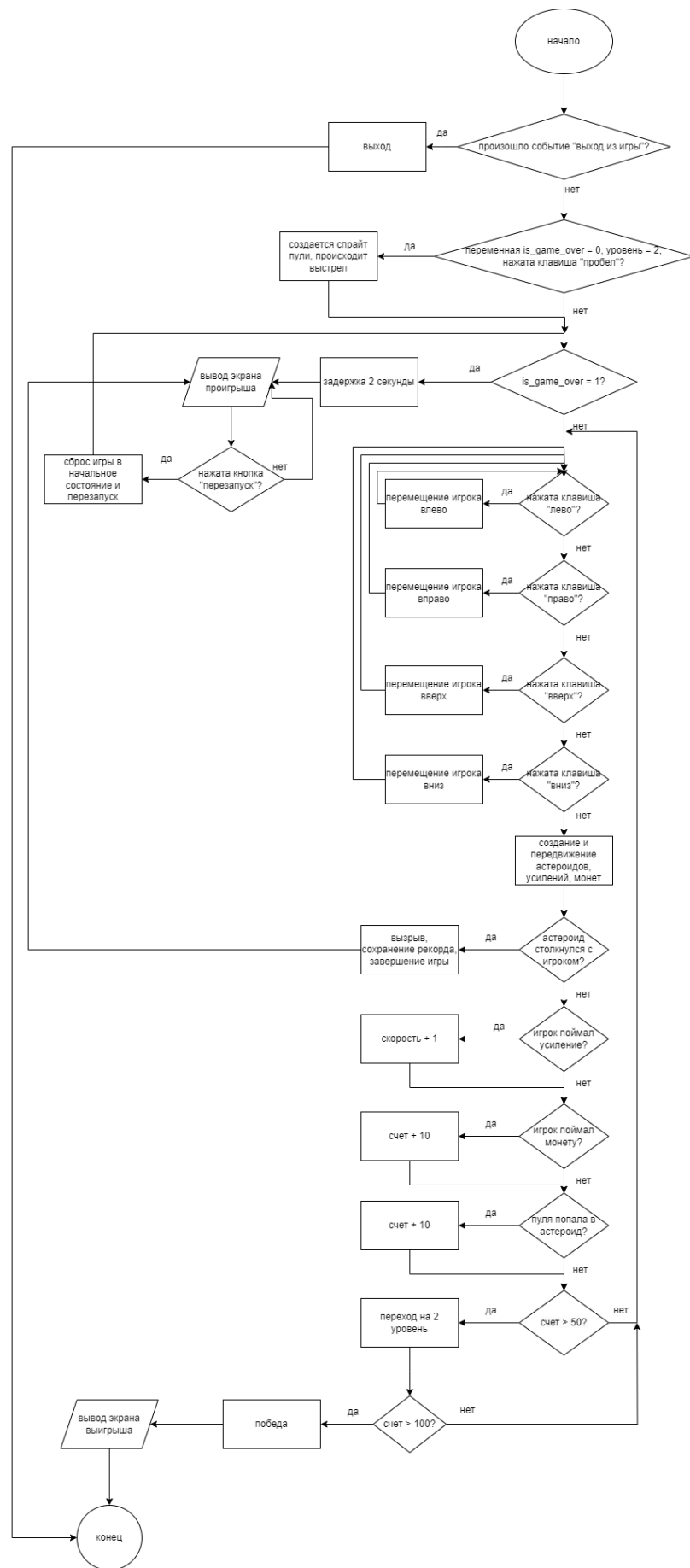


Рисунок 4 – Блок-схема алгоритма игры

### 3.6 Описание функционала программных кодов и представление их скринов

Игра реализована в файле под названием main.py. Импорт необходимых библиотек представлен на рисунке 13.

```
1  import pygame
2  import random
3  import csv
```

Рисунок 13 – Листинг программы

На рисунке 14 показана инициализация игры, заданы начальные и константные переменные для размеров экрана, скорости анимации игрока, максимального значения частоты кадров, счетчика частоты кадров, а также задана переменная для отображения окна экрана и название игры.

```
5  # Initialize pygame
6  pygame.init()
7
8  # Set up the game window
9  SCREEN_WIDTH = 600
10 SCREEN_HEIGHT = 700
11 PLAYER_ANIMATION_SPEED = 10
12 FRAME_RATE = 60
13 frame_counter = 0
14 window = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
15 pygame.display.set_caption("SPACE CAT")
```

Рисунок 14 – Листинг программы

Далее загружаются все используемые изображения для спрайтов и заднего фона игры, а также сразу задается счетчик для анимации персонажа, это отражено на рисунке 15.

```

17 # Load images
18 background_image = pygame.image.load("img/background.png").convert_alpha()
19 player_image = pygame.image.load("img/cat.png").convert_alpha()
20 asteroid_image = pygame.image.load("img/asteroid.png").convert_alpha()
21 powerup_image = pygame.image.load("img/powerup.png").convert_alpha()
22 coin_image = pygame.image.load("img/coin.png").convert_alpha()
23 explosion_image = pygame.image.load("img/explosion.png").convert_alpha()
24 bullet_image = pygame.image.load("img/bullet.png").convert_alpha()
25
26 # Load player animation
27 player_image_animation = [
28     pygame.image.load("img/cat.png").convert_alpha(),
29     pygame.image.load("img/cat 2.png").convert_alpha(),
30     pygame.image.load("img/cat 3.png").convert_alpha(),
31     pygame.image.load("img/cat 4.png").convert_alpha()
32 ]
33 player_image_animation_count = 0

```

Рисунок 15 – Листинг программы

Далее происходит создание классов для каждого спрайта: игрок, астероид, усиление, монета, взрыв, пуля. Все они описаны стандартным для классов образом. Все классы наследуются от класса `pygame.sprite.Sprite` и отвечает за управление движением и положением спрайта на экране. Метод `__init__` является конструктором. Он инициализирует атрибуты класса, такие как изображение, начальное положение и скорость. Метод `update` вызывается в каждом кадре для обновления и регулировки положения спрайта. В целом, каждый класс инкапсулирует атрибуты соответствующего спрайта и поведение движения, что позволяет легко управлять им и обновлять его в игровом цикле. Все описанное отражено на рисунках 16 и 17.

```

36 # Set up the player sprite
37 class Player(pygame.sprite.Sprite):
38     def __init__(self):
39         super().__init__()
40         self.image = player_image
41         self.rect = self.image.get_rect()
42         self.rect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT)
43         self.speed = 5
44         self.speedx = 0
45         self.speedy = 0
46
47     def update(self):
48         self.rect.x += self.speedx
49         self.rect.y += self.speedy
50
51         # Keep the player within the screen boundaries
52         self.rect.clamp_ip(pygame.Rect(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT))
53
54 # Set up the asteroid sprite
55 class Asteroid(pygame.sprite.Sprite):
56     def __init__(self, x, y):
57         super().__init__()
58         self.image = asteroid_image
59         self.rect = self.image.get_rect()
60         self.rect.center = (x, y)
61         self.speed = 5
62
63
64     def update(self):
65         self.rect.y += self.speed
66         if self.rect.top > SCREEN_HEIGHT:
67             self.kill()
68

```

Рисунок 16 – Листинг программы

```

70     # Set up the power-up sprite
71     class Powerup(pygame.sprite.Sprite):
72     def __init__(self):
73         super().__init__()
74         self.image = powerup_image
75         self.rect = self.image.get_rect()
76         self.rect.center = (random.randrange(SCREEN_WIDTH - self.rect.width), -50)
77         self.speed = 4
78
79     def update(self):
80         self.rect.y += self.speed
81         if self.rect.top > SCREEN_HEIGHT:
82             self.kill()
83
84
85     # Set up the coin sprite
86     class Coin(pygame.sprite.Sprite):
87     def __init__(self):
88         super().__init__()
89         self.image = coin_image
90         self.rect = self.image.get_rect()
91         self.rect.center = (random.randrange(SCREEN_WIDTH - self.rect.width), -50)
92         self.speed = 4
93
94     def update(self):
95         self.rect.y += self.speed
96         if self.rect.top > SCREEN_HEIGHT:
97             self.kill()
98
99
100    # Set up the explosion sprite
101    class Explosion(pygame.sprite.Sprite):
102    def __init__(self, x, y):
103        super().__init__()
104        self.image = explosion_image
105        self.rect = self.image.get_rect()
106        self.rect.center = (x, y)
107        self.timer = pygame.time.get_ticks()
108
109    def update(self):
110        if pygame.time.get_ticks() - self.timer > 500:
111            self.kill()
112
113
114    # Set up the bullet sprite
115    class Bullet(pygame.sprite.Sprite):
116    def __init__(self, x, y):
117        super().__init__()
118        self.image = bullet_image
119        self.rect = self.image.get_rect()
120        self.rect.centerx = x
121        self.rect.bottom = y
122        self.speed = -10
123
124    def update(self):
125        self.rect.y += self.speed
126        if self.rect.bottom < 0:
127            self.kill()
128

```

Рисунок 17 – Листинг программы



Все, что касается работы с файлами, отображено на рисунке 18. Функция `save_record`, которая отвечает за добавление новой строки данных в CSV-файл, содержащий записи. Она принимает два параметра: `name` и `score_`. `with open('records.csv', 'a', newline='') as file:` открывает файл с именем 'records.csv' в режиме добавления, что позволяет добавлять новые данные в конец файла. Аргумент `newline=""` используется для обеспечения правильной обработки новых строк при записи в файл. `Writer = csv.writer(file)` создает объект записи из модуля `csv`, который будет использоваться для записи данных в файл. `Writer.writerow([name, score_])` записывает новую строку данных в CSV-файл.

Функция `show_records_table` отвечает за отображение записей из CSV-файла в окне игры. Функция начинается с открытия файла "records.csv" в режиме чтения. Затем он создает объект чтения из модуля `CSV` для чтения содержимого файла. `records = list(reader)` считывает все записи из файла CSV и сохраняет их в списке `records`. Затем функция проверяет количество записей, вычисляя `if len(records) > 9`. Если записей 10, она очищает содержимое файла. После этого функция продолжает рисовать записи в окне игры с помощью `Pygame`. Она начинает с рисования черного прямоугольника во всей области окна с помощью `pygame.draw.rect(window, (0, 0, 0), (0, 0, SCREEN_WIDTH, SCREEN_HEIGHT))`. Переменная `table_spacing` имеет значение 30, что определяет интервал по вертикали между каждой записью в таблице. Затем функция использует функцию `draw_text` для отображения заголовка «Records:». После чего используется цикл для перебора каждой записи в списке «records». Функция `enumerate` используется для получения как индекса (`i`), так и значения (`record`) каждой записи. Для каждой записи вызывается функцию `draw_text`, чтобы отобразить номер индекса, имя и счет в указанных координатах в окне. Наконец, вызывается `pygame.display.flip()` для обновления экрана и отображения записей в окне игры.

```

130     # Open file and append a new line
131     def save_record(name, score_):
132         with open('records.csv', 'a', newline='') as file:
133             writer = csv.writer(file)
134             writer.writerow([name, score_])
135
136     # Display the records from the CSV file
137     def show_records_table():
138         with open("records.csv", "r", newline="") as file:
139             reader = csv.reader(file)
140             records = list(reader)
141
142         # Check the number of records
143         if len(records) > 9:
144             # Clear the file
145             with open("records.csv", "w", newline="") as file:
146                 pass
147
148         pygame.draw.rect(window, (0, 0, 0), (0, 0, SCREEN_WIDTH, SCREEN_HEIGHT))
149         table_spacing = 30
150
151         draw_text(window, "Records:", 32, 300, 10)
152
153         for i, record in enumerate(records):
154             draw_text(window, str(i + 1) + ".", 24, 100, 200 + i * table_spacing)
155             draw_text(window, record[0], 24, 200, 200 + i * table_spacing)
156             draw_text(window, record[1], 24, 500, 200 + i * table_spacing)
157         pygame.display.flip()
158
159
160

```

Рисунок 18 – Листинг программы

Раздел кода на рисунке 19 настраивает игровые часы, инициализирует игровые переменные, создает и добавляет спрайт игрока и различные группы спрайтов.

А также тут отображена вспомогательная функция для вывода текста в окне игры – `draw_text`. Эта функция предназначена для упрощения процесса отображения текста на экране. Она принимает параметры `surface`, `text`, `size`, `x` и `y`, представляющие поверхность (окно игры) для рисования, отображаемый текст, размер шрифта и координаты `x` и `y` положения текста. Она загружает шрифт из файла `"font/candara.ttf"` и отображает текст белым цветом. Визуализированная текстовая поверхность затем позиционируется с помощью `text_rect.midtop = (x, y)`, чтобы центрировать ее по горизонтали и установить

верхнюю позицию, а затем переносится (рисуетя) на указанную поверхность с помощью `surface.blit(text_surface, text_rect)`.

```
137 # Set up the game clock
138 clock = pygame.time.Clock()
139
140 # Set up game variables
141 score = 0
142 speed = 5
143 level = 1
144 is_game_over = False
145 game_over_timer = 0
146
147 # Set up the player sprite
148 player_sprite = Player()
149
150 # Set up the sprite groups
151 asteroid_group = pygame.sprite.Group()
152 powerup_group = pygame.sprite.Group()
153 coin_group = pygame.sprite.Group()
154 explosion_group = pygame.sprite.Group()
155 bullet_group = pygame.sprite.Group()
156 all_sprites = pygame.sprite.Group()
157 all_sprites.add(player_sprite)
158
159
160 # Function to display text on the screen
161 def draw_text(surface, text, size, x, y):
162     font = pygame.font.Font("font/candara.ttf", size)
163     text_surface = font.render(text, True, (255, 255, 255))
164     text_rect = text_surface.get_rect()
165     text_rect.midtop = (x, y)
166     surface.blit(text_surface, text_rect)
```

Рисунок 19 – Листинг программы

Предоставленный на рисунке 20 фрагмент кода определяет отображение меток `label` и функцию `show_game_over_screen`. Метки отображают кнопки "перезапуск" и "рекорды".

Функция `show_game_over_screen()` отвечает за отображение экрана окончания игры. Сначала она рисует фоновое изображение в окне игры, используя `window.blit(background_image, (0, 0))`. Затем он использует вспомогательную функцию `draw_text()` для отображения текста «Game Over», счета игрока и меток для кнопок перезапуска и записи. Метки перезапуска и записи наносятся на поверхность окна в соответствующих прямоугольных

позициях с помощью `window.blit(restart_label, restart_label_rect)` и `window.blit(record_label, record_label_rect)`. Функция также проверяет, была ли нажата метка записи, определяя, находится ли указатель мыши в области `record_label_rect` и нажата ли левая кнопка мыши. Если это так, он вызывает функцию `show_records_table()` для отображения таблицы записей и ждет 2 секунды (`pygame.time.wait(2000)`), прежде чем продолжить. Наконец, отображение обновляется с помощью `pygame.display.flip()`, чтобы отобразить визуализированные элементы на экране.

Функция `show_victory_screen()` отвечает за отображение экрана победы, она действует похожим на предыдущую функцию образом, но выводит лишь текст, сообщающий о победе.

```
193 label = pygame.font.Font("font/candara.ttf", 40)
194
195 # Restart button
196 restart_label = label.render('Restart game', True, (255, 255, 255))
197 restart_label_rect = restart_label.get_rect(center=(SCREEN_WIDTH // 2, SCREEN_HEIGHT - 150))
198
199 # Record button
200 record_label = label.render('Record', True, (255, 255, 255))
201 record_label_rect = record_label.get_rect(center=(SCREEN_WIDTH // 2, SCREEN_HEIGHT - 75))
202
203
204 # Function to show the game over screen
205 def show_game_over_screen():
206     window.blit(background_image, (0, 0))
207     draw_text(window, "Game Over", 64, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 4)
208     draw_text(window, "Score: " + str(score), 40, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2)
209     window.blit(restart_label, restart_label_rect)
210     window.blit(record_label, record_label_rect)
211
212     # Check if the Records button is clicked
213     if record_label_rect.collidepoint(pygame.mouse.get_pos()) and pygame.mouse.get_pressed()[0]:
214         show_records_table()
215         pygame.time.wait(2000)
216
217     pygame.display.flip()
218
219
220 def show_victory_screen():
221     window.blit(background_image, (0, 0))
222     draw_text(window, "YOU WINNER", 64, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2)
223     pygame.display.flip()
224
```

Рисунок 20 – Листинг программы

Предоставленный на рисунке 21 фрагмент кода представляет часть основного игрового цикла игры.

Цикл начинается с перебора событий, возвращаемых функцией `pygame.event.get()`. Если тип события — `pygame.QUIT`, это означает, что пользователь нажал кнопку закрытия окна. В ответ игра завершается, вызывая `pygame.quit()` и `quit()`. Если тип события — `pygame.KEYUP`, он проверяет, не закончилась ли игра (`is_game_over` — `False`), уровень — 2 (`уровень == 2`), а отпущенная клавиша — пробел (`event.key == pygame.K_SPACE`). Если эти условия соблюдены, создается спрайт пули (`Bullet(player_sprite.rect.centerx, player_sprite.rect.top)`) и добавляется в группу `bullet_group`.

Если игра окончена (`is_game_over` имеет значение `True`), код проверяет, прошло ли достаточно времени с момента возникновения события окончания игры. Если разница во времени (`pygame.time.get_ticks()` — `game_over_timer`) превышает 2 секунды, экран завершения игры отображается путем вызова функции `show_game_over_screen()`. Код также проверяет, нажата ли кнопка перезапуска. Если эти условия соблюдены, игра сбрасывается: `is_game_over` устанавливается в `False`, уровень устанавливается в 1, счет сбрасывается в 0, скорость устанавливается в 5, изображение и позиция игрока восстанавливаются, и все спрайты группы опустошены.

Если игра не окончена, код обрабатывает движение игрока в зависимости от нажатых клавиш: влево, вправо, вниз, вверх.

```
226 # Main game loop
227 while True:
228     for event in pygame.event.get():
229         if event.type == pygame.QUIT:
230             pygame.quit()
231             quit()
232         elif event.type == pygame.KEYUP:
233             if is_game_over is False and level == 2 and event.key == pygame.K_SPACE:
234                 bullet_sprite = Bullet(player_sprite.rect.centerx, player_sprite.rect.top)
235                 bullet_group.add(bullet_sprite)
236
237     if is_game_over:
238         if pygame.time.get_ticks() - game_over_timer > 2000:
239             show_game_over_screen()
240             mouse = pygame.mouse.get_pos()
241             # If press restart
242             if restart_label.rect.collidepoint(mouse) and pygame.mouse.get_pressed()[0]:
243                 is_game_over = False
244                 level = 1
245                 score = 0
246                 speed = 5
247                 player_sprite.image = player_image
248                 player_sprite.rect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT)
249                 asteroid_group.empty()
250                 powerup_group.empty()
251                 coin_group.empty()
252                 explosion_group.empty()
253                 bullet_group.empty()
254     else:
255         # Move the player
256         keys = pygame.key.get_pressed()
257         if keys[pygame.K_LEFT]:
258             player_sprite.rect.x -= speed
259         if keys[pygame.K_RIGHT]:
260             player_sprite.rect.x += speed
261         if keys[pygame.K_UP]:
262             player_sprite.rect.y -= speed
263         if keys[pygame.K_DOWN]:
264             player_sprite.rect.y += speed
```

Рисунок 21 – Листинг программы

Приведенный на рисунке 22 фрагмент кода описывает логику создания и обновления игровых объектов, таких как астероиды, бонусы и монеты в игре. Для каждого объекта, используя `random.randint()`, генерируется случайное целое число и если оно меньше или равно заданным числам, создается спрайт со случайной координатой `x` в пределах ширины экрана (`random.randint(0, SCREEN_WIDTH)`) и вверху экрана, а также созданный спрайт добавляется в соответствующую группу.

Для каждого спрайта вызывается метод `update()` для перемещения и обработки любых необходимых обновлений. Если спрайт исчезает с экрана (достигает определенной координаты `y`), он удаляется из своей группы, это помогает поддерживать управляемое количество активных спрайтов в игре.

```
267 # Spawn asteroids randomly
268 if random.randint(1, 400) <= 3:
269     asteroid_sprite = Asteroid(random.randint(0, SCREEN_WIDTH), -50)
270     asteroid_group.add(asteroid_sprite)
271
272 # Move the asteroids and remove them when they go off-screen
273 for asteroid in asteroid_group:
274     asteroid.update()
275
276 # Spawn power-ups randomly
277 if random.randint(1, 400) <= 1:
278     powerup_sprite = Powerup()
279     powerup_group.add(powerup_sprite)
280
281 # Move the power-ups and remove them when they go off-screen
282 for powerup in powerup_group:
283     powerup.update()
284
285 # Spawn coins randomly
286 if random.randint(1, 400) <= 3:
287     coin_sprite = Coin()
288     coin_group.add(coin_sprite)
289
290 # Move the coins and remove them when they go off-screen
291 for coin in coin_group:
292     coin.update()
```

Рисунок 22 – Листинг программы

Предоставленный на рисунке 23 фрагмент кода включает в себя обнаружение столкновений и логику обработки для различных игровых объектов.

Если происходит столкновение игрока и астероида, игрок и участвовавший в столкновении астероид удаляются из соответствующих

групп спрайтов, спрайт взрыва создается в центре позиции игрока с использованием класса «Взрыв» и добавляется в `explosion_group`. Рекорд игрока сохраняется с помощью функции `save_record()` с именем `player` и текущим счетом. Флаг `is_game_over` установлен в `True`, указывая на то, что игра окончена. Переменная `game_over_timer` обновляется текущим временем с помощью `pygame.time.get_ticks()`, чтобы отслеживать отображение игры на экране.

Если происходит столкновение игрока с усилениями, спрайт усиления удаляется из соответствующей группы спрайтов и скорость игрока увеличивается на 1.

Если происходит столкновение игрока с монетами, спрайт монеты удаляется из соответствующей группы спрайтов и счет увеличивается на 10.

При столкновении пули и астероида и пуля, и астероид, участвовавшие в столкновении, удаляются из соответствующих групп спрайтов, а счет игрока увеличивается на 10, что свидетельствует об успешном попадании в астероид.

Метод `update()` вызывается для всех спрайтов в группе `all_sprites`. Этот шаг обновляет позиции, анимацию или другие свойства всех спрайтов в игре.

```
294 # Check for collisions with the player
295 if pygame.sprite.spritecollide(player_sprite, asteroid_group, True):
296     explosion_sprite = Explosion(player_sprite.rect.centerx, player_sprite.rect.centery)
297     explosion_group.add(explosion_sprite)
298
299 # Save the player's record
300 save_record('player', score)
301
302 is_game_over = True
303 game_over_timer = pygame.time.get_ticks()
304
305 # Check for collisions with the power-ups
306 if pygame.sprite.spritecollide(player_sprite, powerup_group, True):
307     speed += 1
308
309 # Check for collisions with the coins
310 if pygame.sprite.spritecollide(player_sprite, coin_group, True):
311     score += 10
312
313 # Move and remove bullets
314 for bullet in bullet_group:
315     bullet.update()
316
317 # Check for collisions between bullets and asteroids
318 collisions = pygame.sprite.groupcollide(bullet_group, asteroid_group, True, True)
319 for bullet, asteroids in collisions.items():
320     score += 10
321
322 # Update all sprites
323 all_sprites.update()
```

Рисунок 23 – Листинг программы

Предоставленный на рисунке 24 фрагмент кода включает логику, связанную с прогрессом уровня и условиями победы. Переход на уровень 2 происходит, когда счет превышает 50. Если счет игрока превышает 100, вызывается функция `show_victory_screen()` для отображения экрана победы. После отображения экрана победы есть задержка в 5 секунд (`pygame.time.wait(5000)`), чтобы игрок мог увидеть сообщение о победе. Наконец, игра завершается вызовом `pygame.quit()` для выхода из модуля Pygame и `quit()` для выхода из программы.

```
# Transition to level 2
if score >= 50 and level == 1:
    level = 2

if score > 100:
    show_victory_screen()
    pygame.time.wait(5000)
    pygame.quit()
    quit()
```

Рисунок 24 – Листинг программы

Предоставленный на рисунке 25 фрагмент кода обрабатывает отрисовку фона игры, спрайта игрока, других спрайтов, счета и информации об уровне с помощью метода `draw()` или `draw_text()` соответственно.

Анимация спрайтов игрока обрабатывается путем проверки списка клавиш на наличие нажатий клавиш. Если нажата любая клавиша со стрелкой, текущий кадр анимации (`player_image_animation_count`) переносится в окно из списка `player_image_animation`. Количество кадров анимации (`player_image_animation_count`) увеличивается на основе счетчика кадров (`frame_counter`) и скорости анимации (`PLAYER_ANIMATION_SPEED`).

Функция `pygame.display.flip()` вызывается для обновления эконога и отображения новой нарисованной графики. Счетчик кадров (`frame_counter`) увеличивается на 1, чтобы контролировать анимацию, если счетчик кадров превышает указанную частоту кадров (`FRAME_RATE`), он сбрасывается на 0.

Функция `clock.tick(60)` вызывается для контроля частоты кадров в игре, ограничивая ее максимум до 60 кадров в секунду.



```

335 # Draw background and sprites
336 window.blit(background_image, (0, 0))
337
338 # Draw the player with animation
339 if keys[pygame.K_LEFT] or keys[pygame.K_RIGHT] or keys[pygame.K_UP] or keys[pygame.K_DOWN]:
340     window.blit(player_image_animation[player_image_animation_count], player_sprite.rect)
341 else:
342     window.blit(player_image, player_sprite.rect)
343
344 if frame_counter % PLAYER_ANIMATION_SPEED == 0: # Check frame counter for animation speed
345     if player_image_animation_count == 3:
346         player_image_animation_count = 0
347     else:
348         player_image_animation_count += 1
349
350 # Draw sprites
351 asteroid_group.draw(window)
352 powerup_group.draw(window)
353 coin_group.draw(window)
354 explosion_group.draw(window)
355 bullet_group.draw(window)
356
357 # Draw score and level information
358 draw_text(window, "Score: " + str(score), 24, SCREEN_WIDTH // 2, 10)
359 draw_text(window, "Level: " + str(level), 24, SCREEN_WIDTH // 2, 40)
360
361 # Update the display
362 pygame.display.flip()
363
364 # Increment frame counter
365 frame_counter += 1
366 if frame_counter >= FRAME_RATE:
367     frame_counter = 0
368
369 # Control the game's frame rate
370 clock.tick(60)

```

Рисунок 25 – Листинг программы

### 3.7 Основные принципы оценки результата завершения игры

Игрок может либо проиграть, не набрав 100 очков, либо выиграть, если количество очков будет больше 100. Очки начисляются за пойманные монеты и уничтоженные астероиды и являются основным критерием для оценки результата завершения игры.

### 3.8 Описание особенностей взаимодействия компьютерных игроков

Разработанная в ходе практики игра предназначена для одного игрока и не предназначена для запуска в мультиплеерном режиме, где могли бы взаимодействовать разные игроки.

## ЗАКЛЮЧЕНИЕ

В ходе тестирования была выявлена проблема. Спрайт игрока выходил за пределы экрана. Это удалось исправить с помощью метода библиотеки Pygame для объекта rect – clamp\_ip. Привязывая положение спрайта игрока к границам игрового экрана, код гарантирует, что игрок не сможет выйти за пределы видимой области. Это предотвращает неожиданное поведение игры.

В процессе исследования теоретических аспектов создания компьютерной игры, были изучены особенности исследования предметной области для разработки компьютерных игр, подходы к формированию интерфейса пользователя и принципы моделирование графических объектов. Описаны различные классификации видеоигр и отличия в подходах крупных студий и инди-разработчиков. Установлено, что языки программирования играют жизненно важную роль в создании компьютерных игр, предоставляя разработчикам средства для воплощения творческих идей. Выбор языка программирования зависит от таких факторов, как требования к производительности, эффективность разработки, совместимость с платформой и конкретные потребности игры.

Рассмотрен язык программирования Python, его история, основные функции и библиотеки, в частности те, которые помогают в разработке игр. Подробно описана использованная библиотека Pygame.

Задачи практической работы направлены на обеспечение всестороннего понимания разработки компьютерных игр с использованием языка программирования Python. Изучая теоретические концепции, анализируя библиотеки для создания игр и приобретая практические навыки, происходит полноценная подготовка для разработки собственной игры.

В результате выполнения практической работы создана игра «SPACE CAT» с помощью библиотеки Pygame языка программирования Python. Использовано объектно-ориентированное программирование (классы, наследование, экземпляры классов), использована системы PIP для установки библиотек, произведена работа с файловой системой для записи сохранений и

результатов игры. Реализована загрузка изображений из файлов, обработка пользовательского ввода и сигналов, различные игровые механики, такие как столкновение и ускорение, использована анимация персонажа и созданы два уровня игры.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Буйначев, С. К., Боклаг, Н. Ю., Песин, Ю. В. Основы программирования на языке Python — Екатеринбург: Изд-во Урал. ун-та, 2014. — 91 с.
- 2 Дроботун Н.В. Алгоритмизация и программирование. Язык Python: учебное пособие. — Санкт-Петербург: Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2020. — 119 с.
- 3 Маккинли, Уэс, Слинкина, А. Python и анализ данных. — Саратов: Профобразование, 2019. — 482 с.