Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Колледж космического машиностроения и технологий

ОТЧЕТ

по производственной практике ПП.01.01 по модулю ПМ.01

«Разработка программных модулей программного обеспечения

для компьютерных систем»

по специальности 09.02.03 «Программирование в компьютерных системах»

Студент 3 курса группы П2-17

Форма обучения: очная

Швыркин Анатолий Павлович

Место прохождения практики

Государственном бюджетном образовательном учреждении высшего образования Московской области «Технологический университет»

(название организации)

Срок прохождения практики с 13 января 2020 г. по 15 марта 2020 г.

Руководители практики

От организации: заведующий мастерской _____ Попов В.Н.

(Должность) (Подпись) (ФИО)

От колледжа: преподаватель _____ Родичкин П.Ф.

(подпись)

Итоговая оценка по практике _____

Оглавление

Введение

На 3 курсе обучения в ККМТ, студентом группы П2-17 Швыркиным Анатолием была проведена производственная практика по модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем». Студент получил задание разработать игру.

Во время прохождения практики я поставил для себя следующие цели:

- Приобрести опыт работы по специальности.
- Закрепить теоретические знания, полученные во время учебы.
- Проанализировать работы отдела.
- Закрепить навыки в разработке проектной и технической документации.
- Закрепить навыки отладки и тестирования программных модулей.

Для выполнения вышеупомянутых мной целей я выдвинул следующие задачи:

- Изучить специфику деятельности организации.
- Установить необходимые инструменты для работы.
- Найти подходящую литературу.

1. Общие сведения о организации.

1.1. Структура организации характеристика основных видов деятельности.

Данное предприятие работает в сфере образования. Университет образован 16 июля 1998 года в форме некоммерческой организации с названием: Негосударственное образовательное учреждение «Королевская академия управления, экономики и социологии».

Технологический университет (ранее Финансово-технологическая академия; Королевский институт управления, экономики и социологии) создан для подготовки кадров новой информации, воспроизводства интеллектуальных ресурсов, формирования инновационных проектов и технологий. Академия находится в наукограде Королеве Московской области – уникальном центре интеллектуальных ресурсов, которые используются для интеграции важнейших знаний и создания систем глобального масштаба.

20 января 2015 года постановлением Правительства Московской области Академии присвоен статус «университета» и вуз переименован в Государственное бюджетное образовательное учреждение высшего образования Московской области «Технологический университет».

Организационная структура колледжа представлена на Рис. 6.1 в Приложении 1.

1.2. Должностные обязанности оператора ЭВМ, техника – программиста, инженера – программиста.

1.2.1. Должностные обязанности оператора ЭВМ.

- осуществляет техническую подготовку документации, необходимой в процессе работы компании. Выполняет копирование документов на ксероксе;
- выполняет набор различных текстов с соблюдением правил орфографии и пунктуации, а также стандартов оформления организационно-распорядительной документации;
- осуществляет работу с электронной почтой, принимает входящие электронные письма и следит за своевременной отправкой исходящих;
- распечатывает и систематизирует нужные документы;
- заносит в компьютерные базы данных различную информацию, важную и необходимую для работы компании;
- следит за состоянием компьютера и копировальной техники;
- своевременно информирует руководство о необходимости приобретения материалов, непосредственно относящихся к производственному процессу.

1.2.2. Должностные обязанности техника – программиста.

- выполняет работу по обеспечению механизированной и автоматизированной обработки, поступающей в ВЦ (ИВЦ) информации, разработки технологии решения экономических и других задач производственного и научно-исследовательского характера;
- принимает участие в проектировании систем обработки данных и систем математического обеспечения машины;
- выполняет подготовительные операции, связанные с осуществлением вычислительного процесса, ведет наблюдение за работой машин;
- составляет простые схемы технологического процесса обработки информации, алгоритмы решения задач, схемы коммутации, макеты, рабочие инструкции и необходимые пояснения к ним;
- разрабатывает программы решения простых задач, проводит их отладку и экспериментальную проверку отдельных этапов работ;
- выполняет работу по подготовке технических носителей информации, обеспечивающих автоматический ввод данных в вычислительную машину, по накоплению и систематизации показателей нормативного и справочного фонда, разработке форм исходящих документов, внесению необходимых изменений и своевременному корректированию рабочих программ;
- участвует в выполнении различных операций технологического процесса обработки информации (прием и контроль входной информации, подготовка исходных данных, обработка информации, выпуск исходящей документации и передача ее заказчику);
- ведет учет использования машинного времени, объемов выполненных работ;
- выполняет отдельные служебные поручения своего непосредственного руководителя.

1.2.3. Должностные обязанности инженера – программиста.

- на основе анализа математических моделей и алгоритмов решения экономических и других задач разрабатывает программы, обеспечивающие возможность выполнения алгоритма и соответственно поставленной задачи средствами вычислительной техники, проводит их тестирование и отладку;
- разрабатывает технологию решения задач по всем этапам обработки информации;

- осуществляет выбор языка программирования для описания алгоритмов и структур данных;
- определяет информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля;
- выполняет работу по подготовке программ к отладке и приводит отладку;
- определяет объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программ их функциональному назначению;
- осуществляет запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач;
- проводит корректировку разработанной программы на основе анализа выходных данных;
- разрабатывает инструкции по работе с программами, оформляет необходимую техническую документацию;
- определяет возможность использования готовых программных продуктов;
- осуществляет сопровождение внедрения программ и программных средств;
- разрабатывает и внедряет системы автоматической проверки правильности программ, типовые и стандартные программные средства, составляет технологию обработки информации;
- выполняет работу по унификации и типизации вычислительных процессов;
- принимает участие в создании каталогов и картотек стандартных программ, в разработке форм документов, подлежащих машинной обработке, в проектировании программ, позволяющих расширить область применения вычислительной техники.

1.3. Основные функции отдела.

- Производственно-технологическая: разработка алгоритма решения задачи на основе предложенной модели; программная реализация алгоритма; отладка и тестирование программных продуктов; модификация программных продуктов; адаптация и настройка программных продуктов; сопровождение программных продуктов; разработка и эксплуатация баз данных; обеспечение достоверности информации при использовании баз данных;

- Организационно-управленческая: организация работы коллектива исполнителей; планирование и организация работ; выбор оптимальных решений при планировании работ в условиях нестандартных ситуаций; участие в оценке качества и экономической эффективности деятельности; обеспечение техники безопасности.

1.4. Документооборот предприятия, структурного подразделения.

Документооборот Отдела в сфере поставленной мне на практике задачи состоит из нескольких этапов:

- получение приказа и распределение работы между сотрудниками;
- перечень существующих дел в Отделе;
- годовой план работ;
- годовой отчет по проделанной работе.

Вид построенной IDEF модели по плану документооборота представлен на рисунках 1 – 3:

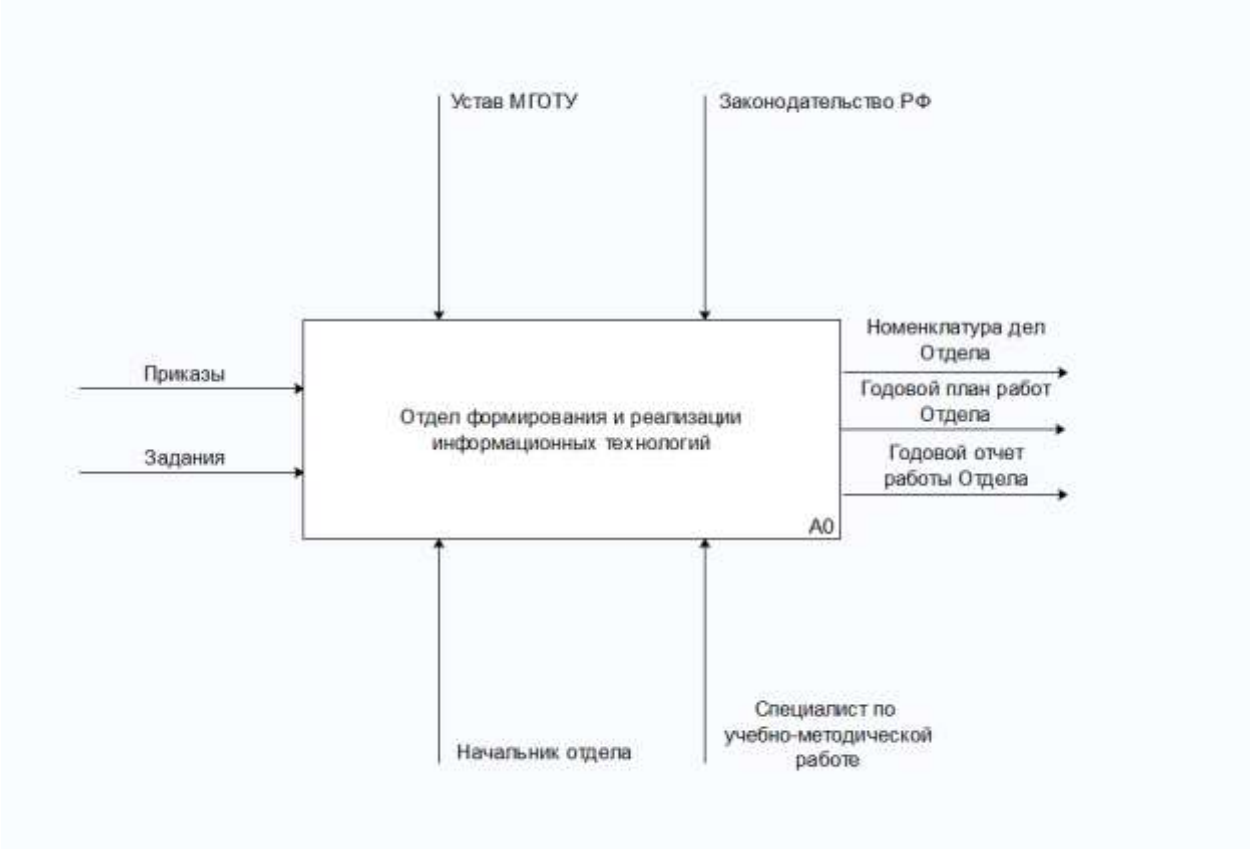

Рис. 1. - IDEF - модель 1 уровень

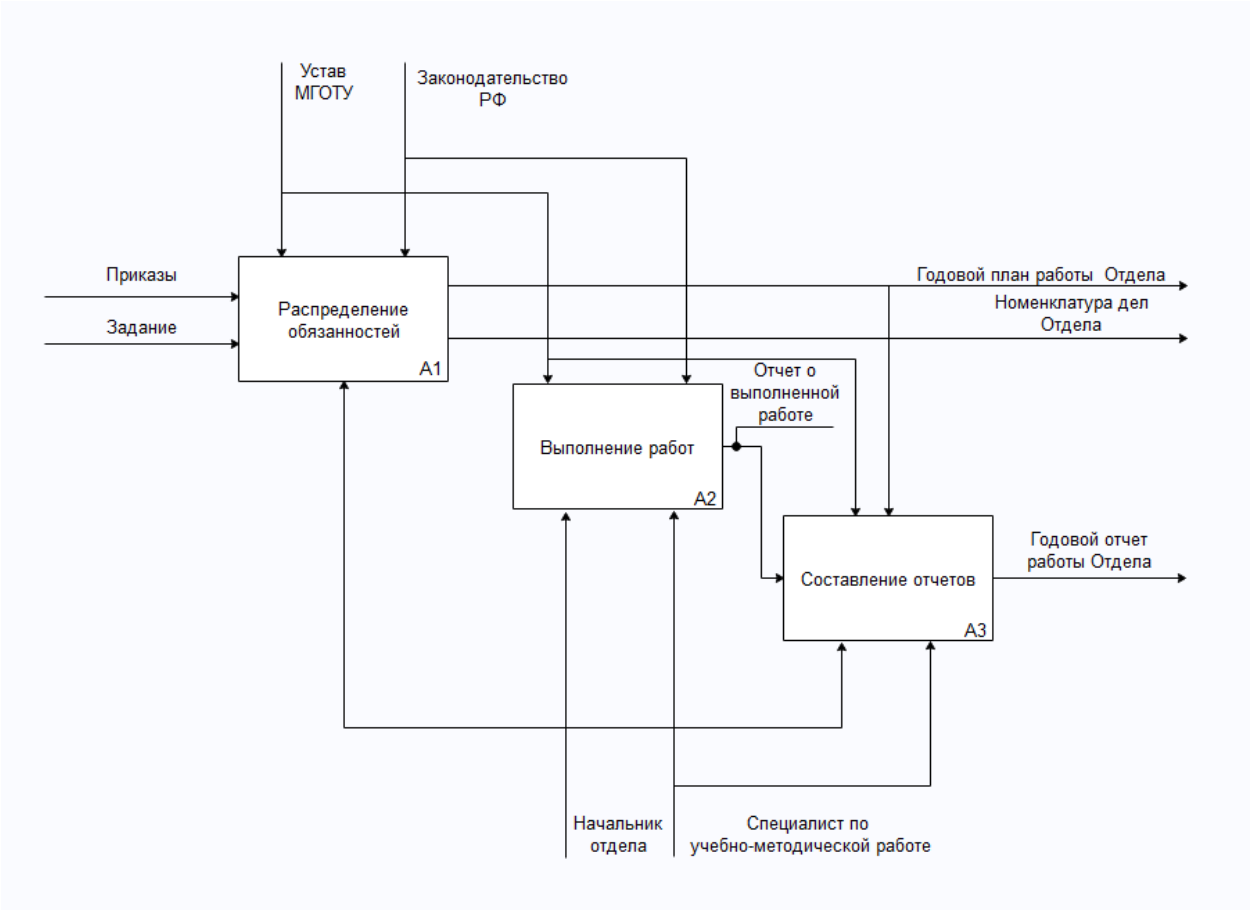

Рис. 2 - IDEF - модель 2 уровень

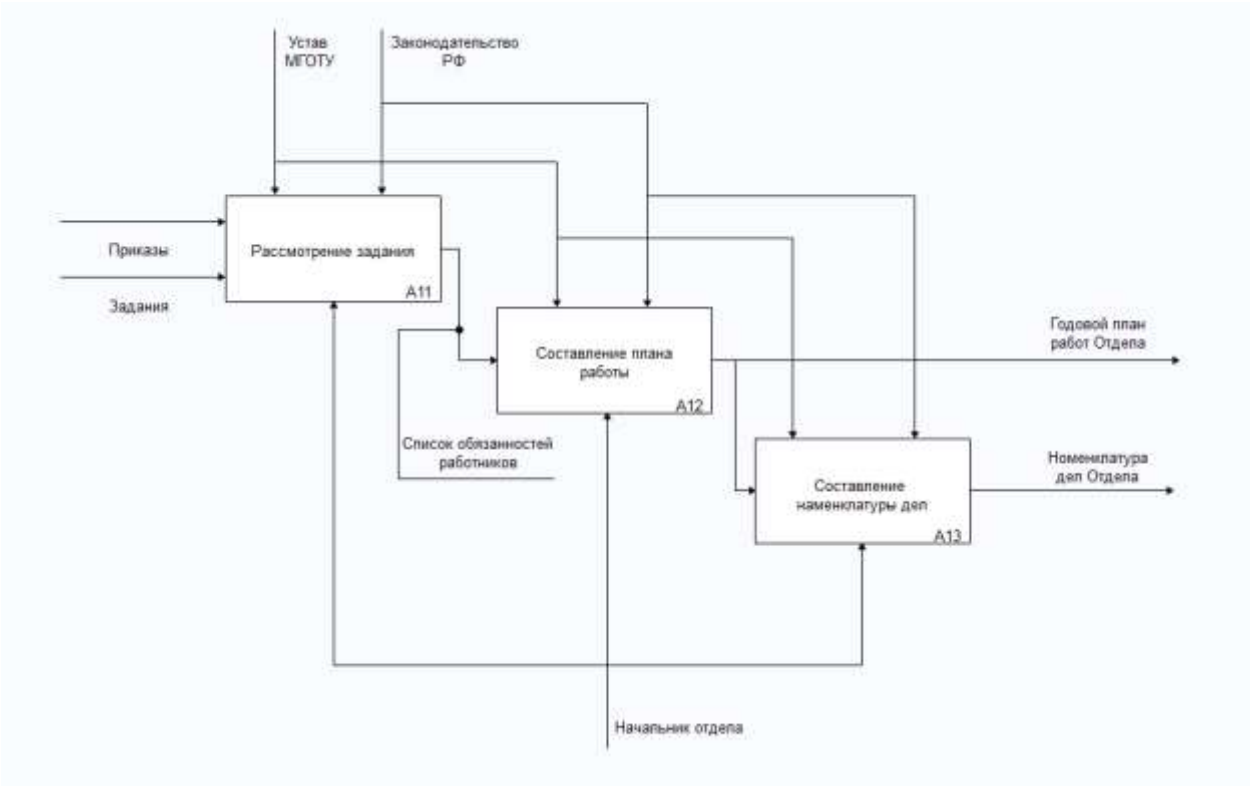

Рис. 3 – IDEF – модель подуровня блока «Распределение задания.

2. Содержание выполняемых видов работ

2.1. Разработка спецификаций отдельный компонентов.

Общее задание было разделено на 3 этапа:

1. Концепция. На основе выданного задания было принято решение писать игру “Вторжение пришельцев”.
2. Написание сценария, по которому будет идти игра.
3. Написание кода игры.

2.2. Коды для игры.

Листинг 1. Основное тело программы.

```
import pygame
from pygame.sprite import Group
from settings import Settings
from game_stats import GameStats
from scoreboard import Scoreboard
from buttom import Button
from ship import Ship
import game_functions as gf

def run_game():
    # Функция Инициализирует pygame, settings и объект экрана.
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode((ai_settings.screen_width,
ai_settings.screen_height))
    pygame.display.set_caption("Alien Invasion")

    # Создание кнопки Play.
    play_button = Button(ai_settings, screen, "Play")

    # Создание экземпляра для хранения игровой статистики.
    stats = GameStats(ai_settings)
    sb = Scoreboard(ai_settings, screen, stats)

    # Создание корабля, группы пуль и группы прищельцев.
    ship = Ship(ai_settings, screen)
    bullets = Group()
    aliens = Group()



    # Создание флота пришельцев.
    gf.create_fleet(ai_settings, screen, ship, aliens)


    # Запуск основного цикла игры.
    while True:
        gf.check_events(ai_settings, screen, stats, sb, play_button, ship,
aliens, bullets)

        if stats.game_active:
```

```
            ship.update()
            gf.update_bullets(ai_settings, screen, stats, sb, ship, aliens,
bullets)
            gf.update_aliens(ai_settings, screen, stats, sb,  ship, aliens,
bullets)

        gf.update_screen(ai_settings, screen, stats, sb, ship, aliens, bullets,
play_button)
run_game()
```

Листинг 2. Модуль отвечающий за работу Корабля.

```
import pygame
from pygame.sprite import Sprite
class Ship(Sprite):
    """Инициализирует корабль и задаёт его начальную позицию."""
    def __init__(self, ai_settings, screen):
        super(Ship, self).__init__()
        self.screen = screen
        self.ai_setting = ai_settings

        # Загрузка изображения корабля и получение прямоугольника.
        self.image = pygame.image.load('images/ship.png')
        self.rect = self.image.get_rect()
        self.screen_rect = screen.get_rect()

        # Каждый новый корабль появляется у нижнего края экрана.
        self.rect.centerx = self.screen_rect.centerx
        self.rect.bottom = self.screen_rect.bottom

        # Сохранение вещественной координаты центра корабля.
        self.center = float(self.rect.centerx)

        # Флаги перемещения.
        self.moving_right = False
        self.moving_left = False
    def update(self):
        """Обновляет позицию корабля с учётом флагов."""
        # Обновляем атрибут center, не rect
        if self.moving_right and self.rect.right < self.screen_rect.right:
            self.center += self.ai_setting.ship_speed_factor
        if self.moving_left and self.rect.left > 0:
            self.center -= self.ai_setting.ship_speed_factor

        # Обновление атрибута rect на основании self.center.
        self.rect.centerx = self.center
    def blitme(self):
        """Рисует корабль в текущей позиции."""
        self.screen.blit(self.image, self.rect)

    def center_ship(self):
        """Размещает корабль в центре нижней стороны."""
        self.center = self.screen_rect.centerx
```

Листинг 3. Модуль отвечающий за работу Пришельца.

```
import pygame
from pygame.sprite import  Sprite

class Alien(Sprite):
    """Класс, представляющий одного пришельца"""

    def __init__(self, ai_settings, screen):
        """Инициализирует пришельца и задаёт его начальную позицию."""
        super(Alien, self).__init__()
        self.screen = screen
        self.ai_settings = ai_settings

        # Загрузка изображения пришельца и назначение атрибута rect.
        self.image = pygame.image.load('images/alien.png')
        self.rect = self.image.get_rect()

        # Каждый новый пришелец появляется в левом верхнем углу экрана.
        self.rect.x = self.rect.width
        self.rect.y - self.rect.height

        # Сохранение точной позиции пришельца.
        self.x = float(self.rect.x)

    def blitme(self):
        """Выводит пришельца в текущем положении."""
        self.screen.blit(self.image, self.rect)

    def check_edges(self):
        """Возвращает True, если пришелец находится у края экрана."""
        screen_rect = self.screen.get_rect()
        if self.rect.right >= screen_rect.right:
            return True
        elif self.rect.left <= 0:
            return True

    def update(self):
        """Перемещает пришельцев влево или вправо."""
        self.x += (self.ai_settings.alien_speed_factor *
self.ai_settings.fleet_direction)
        self.rect.x = self.x
```

Листинг 4. Модуль отвечающий за все игровые механики.

```
import sys
from time import sleep
import pygame
from bullet import Bullet
from alien import Alien

def check_keydown_events(event, ai_settings, screen, ship, bullets):
    """Реагирует на нажатие клавиш."""
    if event.key == pygame.K_RIGHT:
        ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        ship.moving_left = True
```

```
    elif event.key == pygame.K_SPACE:
        fire_bullet(ai_settings, screen, ship, bullets)
    elif event.key == pygame.K_q:
        sys.exit()

def check_keyup_events(event, ship):
    """Реагирует на отпускание клавиш."""
    if event.key == pygame.K_RIGHT:
        ship.moving_right = False
    elif event.key == pygame.K_LEFT:
        ship.moving_left = False

def check_events(ai_settings, screen, stats, sb, play_button, ship, aliens,
bullets):
    """Обрабатывает нажатие клавиш и события мыши."""
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            check_keydown_events(event, ai_settings, screen, ship, bullets)
        elif event.type == pygame.KEYUP:
            check_keyup_events(event, ship)
        elif event.type == pygame.MOUSEBUTTONDOWN:
            mouse_x, mouse_y = pygame.mouse.get_pos()
            check_play_button(ai_settings, screen, stats, sb, play_button,
ship, aliens, bullets, mouse_x, mouse_y)

def check_play_button(ai_settings, screen, stats, sb, play_button, ship,
aliens, bullets, mouse_x, mouse_y):
    """Запускает новую игру при нажатии кнопки Play."""
    button_clicked = play_button.rect.collidepoint(mouse_x, mouse_y)
    if button_clicked and not stats.game_active:
        # Сброс игровых настроек.
        ai_settings.initialize_dynamic_settings()

        # Скрытие курсора мыши.
        pygame.mouse.set_visible(False)

        # Сброс игровой статистики.
        stats.reset_stats()
        stats.game_active = True

        # Перезапуск счётчика очков.
        sb.prep_score()
        sb.prep_high_score()
        sb.prep_level()
        sb.prep_ships()

        # Отчистка списков пришельцев и пуль.
        aliens.empty()
        bullets.empty()

        # Создание нового флота и размещение корабля в центре.
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
def fire_bullet(ai_settings, screen, ship, bullets):
```

```
    """Прорисовка пуль, если лимит не достигнут."""
    # Создание новой пули и добавление в список пуль.
    if len(bullets) < ai_settings.bullets_allowed:
        new_bullet = Bullet(ai_settings, screen, ship)
        bullets.add(new_bullet)

def update_screen(ai_settings, screen, stats, sb, ship, aliens, bullets,
play_button):
    """Обновляет позиции пуль и уничтожает старые пули."""
    screen.fill(ai_settings.bg_color)

    # Перерисовать все пули, позади корабля и пришельцев.
    for bullet in bullets.sprites():
        bullet.draw_bullet()
    ship.blitme()
    aliens.draw(screen)

    # Нарисовать информацию о счете.
    sb.show_score()

    # Нарисовать кнопку воспроизведения, если игра неактивна.
    if not stats.game_active:
        play_button.draw_button()

    # Сделать последний нарисованный экран видимым.
    pygame.display.flip()

def update_bullets(ai_settings, screen, stats, sb, ship, aliens, bullets):
    """Обновляет положение пуль и удаляет старые."""
    # Обновление позиции пули.
    bullets.update()

    # Избавлени от исчезнувших пуль.
    for bullet in bullets.copy():
        if bullet.rect.bottom <= 0:
            bullets.remove(bullet)

    check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship, aliens,
bullets)

def check_high_score(stats, sb):
    """Проверяет, установлен ли новый рекорд"""
    if stats.score > stats.high_score:
        stats.high_score = stats.score
        sb.prep_high_score()

def check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship, aliens,
bullets):
    """Реагирует на столкновения пули с пришельцами."""
    # Удалить все пули и пришельцев, которые столкнулись.
    collisions = pygame.sprite.groupcollide(bullets, aliens, True, True)

    if collisions:
        for aliens in collisions.values():
            stats.score += ai_settings.alien_points * len(aliens)
            sb.prep_score()
```

```
        check_high_score(stats, sb)

    if len(aliens) == 0:
        # Если весь флот уничтожен, начать новый уровень.
        bullets.empty()
        ai_settings.increase_speed()

        # Увелничение уровня.
        stats.level += 1
        sb.prep_level()

        create_fleet(ai_settings, screen, ship, aliens)

def check_fleet_edges(ai_settings, aliens):
    """Отслеживает если какой либо из пришельцев достиг границы."""
    for alien in aliens.sprites():
        if alien.check_edges():
            change_fleet_direction(ai_settings, aliens)
            break

def change_fleet_direction(ai_settings, aliens):
    """Смещение флота и изменение направления."""
    for alien in aliens.sprites():
        alien.rect.y += ai_settings.fleet_drop_speed
    ai_settings.fleet_direction *= -1

def ship_hit(ai_settings, screen, stats, sb, ship, aliens, bullets):
    """Реагирует на попадание кораблся по пришельцу."""
    if stats.ships_left > 0:
        # Смещение корабля влево.
        stats.ships_left -= 1

        # Обновление счётчика очков.
        sb.prep_ships()

    else:
        stats.game_active = False
        pygame.mouse.set_visible(True)

    # Очистить список инопланетян и пуль.
    aliens.empty()
    bullets.empty()

    # Создаёт новый флот и центрирует корабль.
    create_fleet(ai_settings, screen, ship, aliens)
    ship.center_ship()

    # Пауза.
    sleep(0.5)

def check_aliens_bottom(ai_settings, screen, stats, sb, ship, aliens, bullets):
    """Проверяет, достигли ли инопланетяне нижней части экрана."""
    screen_rect = screen.get_rect()
    for alien in aliens.sprites():
        if alien.rect.bottom >= screen_rect.bottom:
            ship_hit(ai_settings, screen, stats, sb, ship, aliens, bullets)
```

```
            break

def update_aliens(ai_settings, screen, stats, sb, ship, aliens, bullets):
    """Проверяет, находится ли флот на границе экрана, затем обновите позиции
всех инопланетян во флоте."""
    check_fleet_edges(ai_settings, aliens)
    aliens.update()

    # Ищите столкновения инопланетян и корабля.
    if pygame.sprite.spritecollideany(ship, aliens):
        ship_hit(ai_settings, screen, stats, sb, ship, aliens, bullets)

    # Ищет инопланетян, дошедших до нижней части экрана.
    check_aliens_bottom(ai_settings, screen, stats, sb, ship, aliens, bullets)

def get_number_aliens_x(ai_settings, alien_width):
    """Определяет количество пришельцев в ряду"""
    available_space_x = ai_settings.screen_width - 2 * alien_width
    number_aliens_x = int(available_space_x / (2 * alien_width))
    return number_aliens_x

def get_number_rows(ai_settings, ship_height, alien_height):
    """Определяет количество рядов пришельцев, которые помещяются на экране"""
    available_space_y = ai_settings.screen_height - (3 * alien_height) -
ship_height
    number_rows = int(available_space_y / (2 * alien_height))
    return number_rows

def create_alien(ai_settings, screen, aliens, alien_number, row_number):
    """Создаёт пришелеьца, и ставит его в ряд."""
    alien = Alien(ai_settings, screen)
    alien_width = alien.rect.width
    alien.x = alien_width + 2 * alien_width * alien_number
    alien.rect.x = alien.x
    alien.rect.y = alien.rect.height + 2 * alien.rect.height * row_number
    aliens.add(alien)

def create_fleet(ai_settings, screen, ship, aliens):
    """Создаёт полный флот пришельцев."""
    # Создаёт пришельца, и заносит его номер в список.
    alien = Alien(ai_settings, screen)
    number_aliens_x = get_number_aliens_x(ai_settings, alien.rect.width)
    number_rows = get_number_rows(ai_settings, ship.rect.height,
alien.rect.height)

    # Создания флота пришельцев.
    for row_number in range(number_rows):
        for alien_number in range(number_aliens_x):
            create_alien(ai_settings, screen, aliens, alien_number, row_number)
```

Листинг 5. Модуль отвечающий прорисовку пуль.

```
import pygame
from pygame.sprite import Sprite
```

```
class Bullet(Sprite):
    """Класс для управления пулями, выпушенными кораблём."""
    def __init__(self, ai_settings, screen, ship):
        """Создаём объект пули в текущей позиции корабля."""
        super(Bullet, self).__init__()
        self.screen = screen

        # Создание пули в позиции (0,0) и назначение правильной позиции.
        self.rect = pygame.Rect(0, 0, ai_settings.bullet_width,
                                ai_settings.bullet_height)
        self.rect.centerx = ship.rect.centerx
        self.rect.top = ship.rect.top

        # Позиция пули хранится в вещественном формате.
        self.y = float(self.rect.y)

        self.color = ai_settings.bullet_color
        self.speed_factor = ai_settings.bullet_speed_factor

    def update(self):
        """Перемещает пулю вверх по экрану."""
        # Обновление позиции пули в вещественном формате.
        self.y -= self.speed_factor
        # Обновление позиции прямоугольника.
        self.rect.y = self.y

    def draw_bullet(self):
        """Вывод пули на экран."""
        pygame.draw.rect(self.screen, self.color, self.rect)
```

Листинг 6. Модуль отвечающий за кнопку “Play” в игре.

```
import pygame.font

class Button():
    def __init__(self, ai_settings, screen, msg):
        """Инициализирует атрибуты конпки."""
        self.screen = screen
        self.screen_rect = screen.get_rect()

        # Назначение размеров и свойств кнопок.
        self.width, self.height = 200, 50
        self.button_color = (0, 255, 0)
        self.text_color = (255, 255, 255)
        self.font = pygame.font.SysFont(None, 48)

        # Построение объекта rect кнопки и выравнивание по центру экрана.
        self.rect = pygame.Rect(0, 0, self.width, self.height)
        self.rect.center = self.screen_rect.center

        # Сообщение кнопки создаётся только один раз.
        self.prep_msg(msg)

    def prep_msg(self, msg):
        """Преобразует msg в прямоугольник и выравнивает текст по центру."""
        self.msg_image = self.font.render(msg, True, self.text_color,
```

```
self.button_color)
        self.msg_image_rect = self.msg_image.get_rect()
        self.msg_image_rect.center = self.rect.center

    def draw_button(self):
        # Отображение пустой кнопки и вывод сообщения.
        self.screen.fill(self.button_color, self.rect)
        self.screen.blit(self.msg_image, self.msg_image_rect)
```

Листинг 7. Модуль игровой статистики.

```
class GameStats():
    """Отслеживание статистики для игры Alien Invasion."""
    def __init__(self, ai_settings):
        """Инициализирует статистику."""
        self.ai_settings = ai_settings
        self.reset_stats()

        # Игра Alien Invasion запускается в неактивном состоянии.
        self.game_active = False
    def reset_stats(self):
        """Инициализирует статистику, изменяющуюся в ходе игры."""
        self.ships_left = self.ai_settings.ship_limit
        self.score = 0
        self.level = 1

        # Рекорд не должен сбрасываться.
        self.high_score = 0
```

Листинг 8. Модуль игрового счёта.

```
import pygame.font
from pygame.sprite import Group

from ship import Ship


class Scoreboard:
    """Сообщает информацию о баллах."""
    def __init__(self, ai_settings, screen, stats):
        """Инициализирует атрибуты ведения счета."""
        self.screen = screen
        self.screen_rect = screen.get_rect()
        self.ai_settings = ai_settings
        self.stats = stats

        # Настройки шрифта для скоринга информации.
        self.text_color = (30, 30, 30)
        self.font = pygame.font.SysFont(None, 48)

        # Подготовить начальное количество очков.
        self.prep_score()
        self.prep_high_score()
        self.prep_level()
        self.prep_ships()

    def prep_score(self):
        """Превращает очки в визуализированное изображение."""
```

```
        rounded_score = int(round(self.stats.score, -1))
        score_str = "{:,}".format(rounded_score)
        self.score_image = self.font.render(score_str, True, self.text_color,
self.ai_settings.bg_color)

        # Показать счет в правом верхнем углу экрана.
        self.score_rect = self.score_image.get_rect()
        self.score_rect.right = self.screen_rect.right - 20
        self.score_rect.top = 20

    def prep_high_score(self):
        """Превращает рекорд в визуализированное изображение."""
        high_score = int(round(self.stats.high_score, -1))
        high_score_str = "{:,}".format(high_score)
        self.high_score_image = self.font.render(high_score_str, True,
self.text_color, self.ai_settings.bg_color)

        # Отображает рекорд в центре в верхней части экрана.
        self.high_score_rect = self.high_score_image.get_rect()
        self.high_score_rect.centerx = self.screen_rect.centerx
        self.high_score_rect.top = self.score_rect.top

    def prep_level(self):
        """Превращает уровень в визуализированное изображение."""
        self.level_image = self.font.render(str(self.stats.level), True,
self.text_color, self.ai_settings.bg_color)

        # Поместить уровень ниже количества очков.
        self.level_rect = self.level_image.get_rect()
        self.level_rect.right = self.score_rect.right
        self.level_rect.top = self.score_rect.bottom + 10

    def prep_ships(self):
        """Показывает, сколько кораблей осталось."""
        self.ships = Group()
        for ship_number in range(self.stats.ships_left):
            ship = Ship(self.ai_settings, self.screen)
            ship.rect.x = 10 + ship_number * ship.rect.width
            ship.rect.y = 10
            self.ships.add(ship)

    def show_score(self):
        """Отображения счёта на экране."""
        self.screen.blit(self.score_image, self.score_rect)
        self.screen.blit(self.high_score_image, self.high_score_rect)
        self.screen.blit(self.level_image, self.level_rect)
        # Отображение кораблей.
        self.ships.draw(self.screen)
```

Листинг 9. Модуль игровых настроек.

```
class Settings():
    """Класс для хранения всех настроек игры Alien Invasion,"""

    def __init__(self):
        """Инициализирует статические настройки игры."""
```

```
        # Параметры экрана
        self.screen_width = 1600
        self.screen_height = 900
        self.bg_color = (185, 185, 185)

        # Настройки корабля
        self.ship_speed_factor = 1
        self.ship_limit = 3

        # Параметры пули.
        self.bullet_speed_factor = 3
        self.bullet_width = 3
        self.bullet_height = 15
        self.bullet_color = 60, 60, 60
        self.bullets_allowed = 3

        # Настройки пришельцев
        self.fleet_drop_speed = 10


        # Темп ускорения игры.
        self.speedup_scale = 1.1
        self.initialize_dynamic_settings()

        # Темп роста стоимости пришельцев.
        self.score_scale = 1.5
        self.initialize_dynamic_settings()

    def initialize_dynamic_settings(self):
        """Инициализирует настройки, изменяющиеся в ходе игры."""
        self.ship_speed_factor = 1.5
        self.bullet_speed_factor = 3
        self.alien_speed_factor = 1

        # fleet_direction = 1 обозначает движение вправо; а -1 влево.
        self.fleet_direction = 1

        # Подсчёт очков.
        self.alien_points = 50

    def increase_speed(self):
        """Увеличивает настройки скорости и стоимости прищельцев."""
        self.ship_speed_factor *= self.speedup_scale
        self.bullet_speed_factor *= self.speedup_scale
        self.alien_speed_factor *= self.speedup_scale
        self.alien_points = int(self.alien_points * self.score_scale)
        print(self.alien_points)
```

3. Обращение с программой

Ввиду особенностей дополнительной библиотеки Pygame, конвертация в формат .exe не представляется возможной.

Для запуска программы необходимо выполнить следующие действия:

Иметь в наличии на системной машине:

- Python 3
- Библиотеки установленные с помощью pip: Pygame, pillow, wheel.
- Среда разработки IDLE (устанавливается вместе с Python 3) или Pycharm

Необходимо выполнить следующие действия:

- Распаковать папку Alien_Invasion с диска с программой.
- Запустить файл alien_invasion.py.

4. Выводы

Полученные навыки:

- Работа с компонентами Python и Windows PowerShell.
- Использование дополнительных библиотек Pygame.
- Проектирование и создание основных механик.

Полученные умения:

- Работа в PyCharm Community Edition
- Работа с модулем Pygame

5. Заключение

Перед прохождением производственной практики в Государственном бюджетном образовательном учреждении высшего образования Московской области «Технологический университет» мной были поставлены следующие основные цели:

- Приобрести опыт работы по специальности.
- Закрепить теоретические знания, полученные во время учебы.
- Выполнение требований и действий, предусмотренных программой производственной практики и заданий руководителя.
- Закрепить навыки в разработке проектной и технической документации.
- Закрепить навыки отладки и тестирования программных модулей.

По окончанию практики я добился, чего хотел. Для достижения своих целей я использовала интернет – источники, документацию средств разработки.

Во время прохождения практики я приобрел опыт работы по специальности. Так же был закреплен навык разработки проектной и технической документации и навык отладки и тестирования программных модулей.

По окончанию практики был составлен отчёт.

6. Дневник практики

Таблица 1. Дневник практики.

Дата	Содержание работы	Отметка о выполнении работы	Подпись руководителя практики
13.01 – 22.01	Выбор языка программирования, выбор среды разработки, выбор программного обеспечения.		
23.01 – 3.02	Создание базовых механик для игры.		
4.02 – 19.02	Работа с Pygame. Создание игрового пространства, скачивание дополнительных библиотек.		
20.02 – 26.02	Создание игровых объектов и дальнейшая работа с ними		
27.02 – 09.03	Рефакторинг и отладка		
10.03 – 15.03	Оформление отчёта. Обработка листингов, группировка отчёта.		

7. Список использованной литературы.

1. https://docs.python.org/3/
2. https://github.com/pygame/
3. https://www.pygame.org/project/1039

8. Приложения.

Приложение 1.

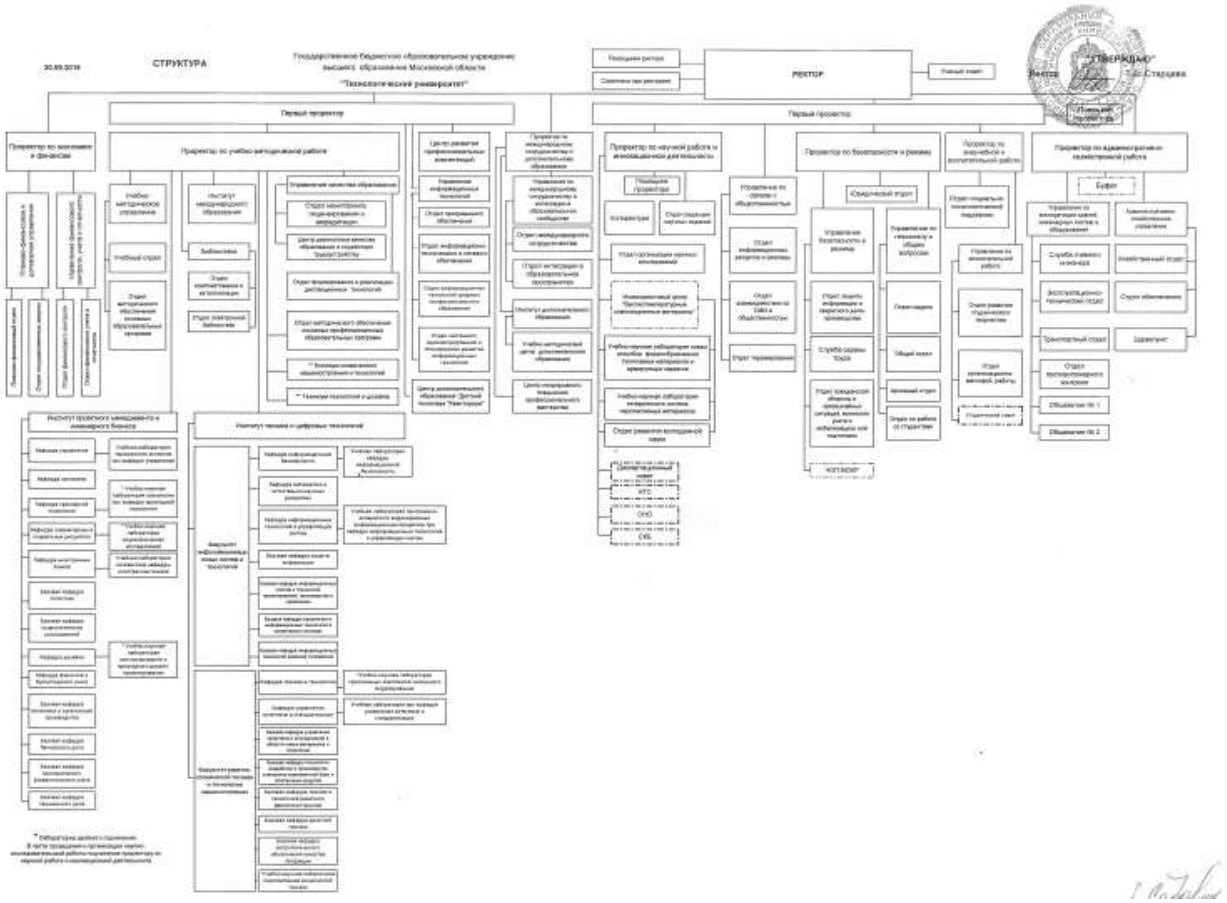

Рис. 6.1. Организационная структура колледжа