

**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»

Выполнил студент гр. П2-18

Короленко Иван Романович

\_\_\_\_\_\_\_\_\_\_ (подпись)

Принял преподаватель

Попов В.Н.

\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_(оценка)

**Оглавление**

[**1.** **Общие сведения о организации.** 4](#_Toc73708685)

[**1.1.** **Структура организации характеристика основных видов деятельности**. 4](#_Toc73708686)

[**1.2.** **Должностные обязанности оператора ЭВМ, техника – программиста, инженера – программиста.** 4](#_Toc73708687)

[**1.2.1.** **Должностные обязанности оператора ЭВМ.** 4](#_Toc73708688)

[**1.2.2.** **Должностные обязанности техника – программиста.** 5](#_Toc73708689)

[**1.2.3.** **Должностные обязанности инженера – программиста.** 6](#_Toc73708690)

[**1.3.** **Основные функции отдела.** 8](#_Toc73708691)

[**1.4.** **Документооборот предприятия, структурного подразделения.** 8](#_Toc73708692)

[**2.** **Содержание выполняемых видов работ** 12](#_Toc73708693)

[**2.1.** **Разработка спецификаций отдельный компонентов**. 12](#_Toc73708694)

[**2.2.** **Скрипты для игры.** 12](#_Toc73708695)

[**1.1.** **Отладка.** 19](#_Toc73708696)

[**1.2.** **Тестирование программы.** 19](#_Toc73708697)

[**1.3.** **Оптимизация программного кода.** 19](#_Toc73708698)

[**3.** **Выводы** 20](#_Toc73708699)

[**4.** **Дневник практики** 21](#_Toc73708700)

[**5.** **Список использованной литературы.** 22](#_Toc73708701)

[**6.** **Приложения.** 23](#_Toc73708702)

**Введение**

На 3 курсе обучения в ККМТ, студентом группы П2-18 Короленко Иваном была пройдена производственная практика по модулю ПМ.01  
«Разработка программных модулей программного обеспечения   
для компьютерных систем». Студент получил задание разработать игру.

Во время прохождения практики 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.

# **Общие сведения о организации.**

## **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 формации, 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 Рисунок 2 в Приложении 1.

## **Должностные обязанности оператора ЭВМ, техника – программиста, инженера – программиста.**

### **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.

### **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.

### **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.

## **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.

## **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.

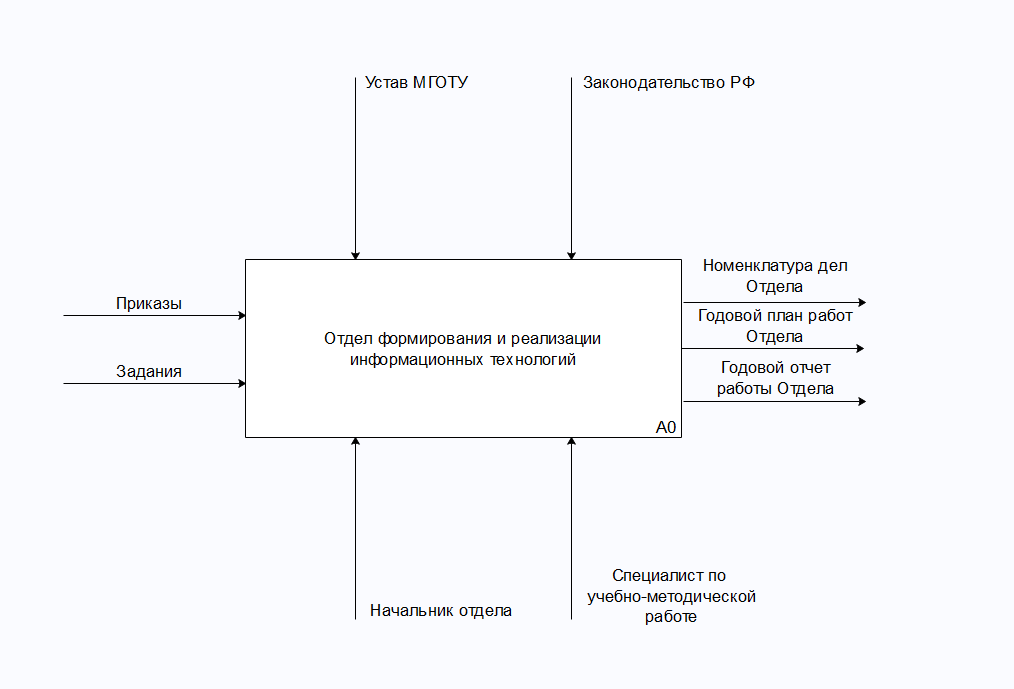
Вид построенной IDEF модели по плану документооборота представлен на рисунках 1–3:

Рисунок 1 IDEF - модель 1 уровень

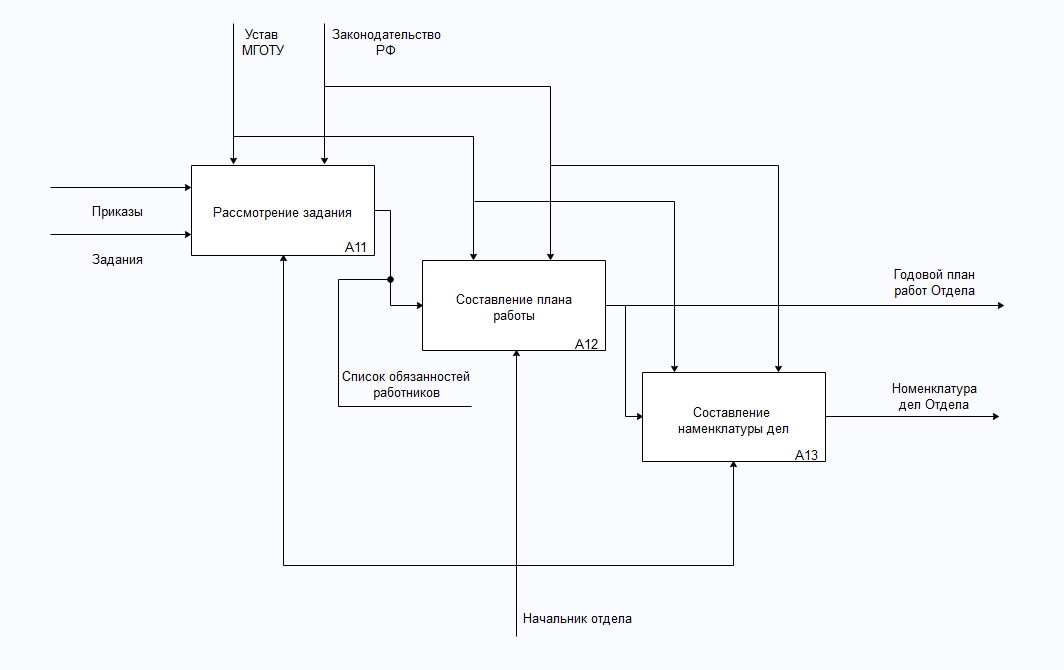


Рисунок 2 - IDEF - модель 2 уровень

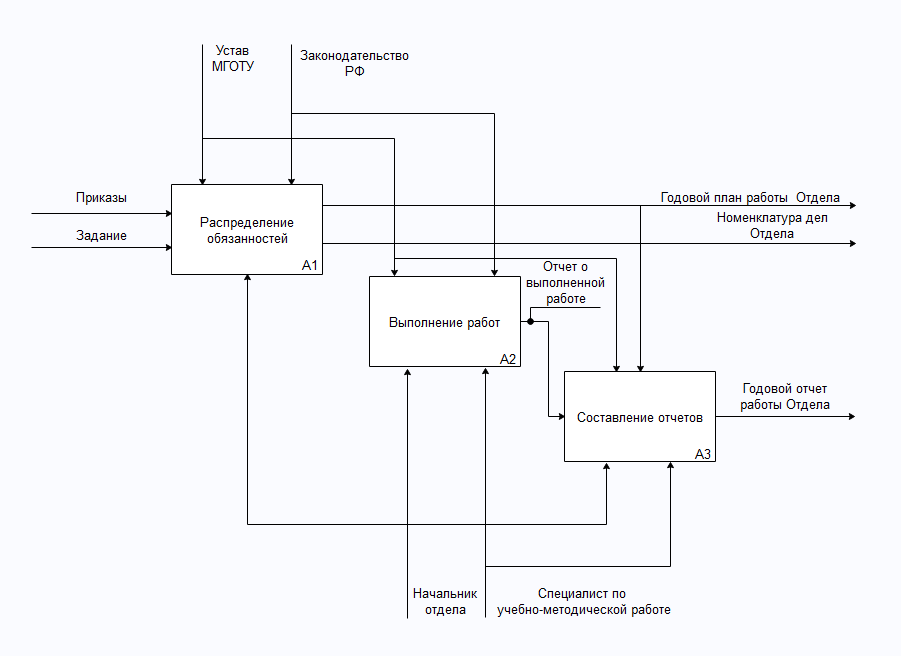


Рисунок 3 - IDEF – модель подуровня блока «Распределение задания»

# **Содержание выполняемых видов работ**

## **Разработка спецификаций отдельный компонентов**.

Общее задание было разделено на 3 этапа:

1. Рисование объектов, и настройка арены.
2. Реализация игровых механизмов.
3. Скрипты.

## **Скрипты для игры.**

**Листинг 1. Основной скрипт двигателя и опорного механизма.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class LogRotation : MonoBehaviour

{

[System.Serializable] //Это позволит редактировать в редакторе

//Настраиваемый класс, позволяющий изменять скорость вращение "Log"

private class RotationElement

{

//pragma warning - избавление от ошибок

#pragma warning disable 0649

public float Speed;

public float Duration;

#pragma warning restore 0649

}

[SerializeField] //Делает приватные поля доступными для редактирования

//Образец полного вращения "Log"

private RotationElement[] rotationPattern;

//Устанавливает Wheel Joint 2D из объекта "LogMotor"

private WheelJoint2D wheelJoint;

//Сила, прикладываемая к бревну через Wheel Joint 2D

private JointMotor2D motor;

private void Awake()

{

//Поля настройки

wheelJoint = GetComponent<WheelJoint2D>();

motor = new JointMotor2D();

//Запуск доп. программы с бесконечным циклом, когда этот скрипт пробуждается

StartCoroutine("PlayRotationPattern")

}

private IEnumerator PlayRotationPattern()

{

int rotationIndex = 0;

//Вышеупомянутый бесконечный цикл доп. программы

while (true)

{

//Работа с физикой, так же, как и (FixedUpdate method)

yield return new WaitForFixedUpdate();

motor.motorSpeed = rotationPattern[rotationIndex].Speed;

motor.maxMotorTorque = 10000; //Крутящий момент

wheelJoint.motor = motor; //Обновление Motor

//Даём движку делать своё дело в течение указанного времени

yield return new WaitForSecondsRealtime(rotationPattern[rotationIndex].Duration);

rotationIndex++;

//Бесконечный цикл через ротацию

rotationIndex = rotationIndex < rotationPattern.Length ? rotationIndex : 0;

}

}

}

**Листинг 2. Скрипт ножа.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Knife : MonoBehaviour

{

[SerializeField]

private Vector2 throwForce;

//Нож не активен, до тех пор, пока не управляет игрок

//В тот момент, когда нож попадает в бревно, то он создаёт новый

private bool isActive = true;

private Rigidbody2D rb; //Управление физикой

private BoxCollider2D knifeCollider; //коллайдер, прикрепленный к Knife

private void Awake()

{

rb = GetComponent<Rigidbody2D>();

knifeCollider = GetComponent<BoxCollider2D>();

}

void Update()

{

//Этот метод определения ввода также работает на касание

if (Input.GetMouseButtonDown(0) && isActive)

{

rb.AddForce(throwForce, ForceMode2D.Impulse); //Бросок ножа

//Как только нож не стоит на месте, мы можем применить силу тяжести (он не упадет автоматически)

rb.gravityScale = 1;

//Делает кол-во ножей меньше

GameController.Instance.GameUI.DecrementDisplayedKnifeCount();

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

//Проверка на столкновение, когда нож активен

if (!isActive)

return;

//Когда нож активен, то первое столкновение деактивируем его

isActive = false;

//Условие столкновения с бревном

if (collision.collider.tag == "Log")

{

//Воспроизводим эффект частицы при столкновении

GetComponent<ParticleSystem>().Play();

rb.velocity = new Vector2(0, 0);//Тормоз ножа

//Автоматически унаследует ротацию нового родителя

rb.bodyType = RigidbodyType2D.Kinematic;

transform.SetParent(collision.collider.transform);

//Перемещение в зависимотси от лезвия, которое застряло в бревне

knifeCollider.offset = new Vector2(knifeCollider.offset.x, -0.23f);

knifeCollider.size = new Vector2(knifeCollider.size.x, 0.3f);

//Создание нового ножа

GameController.Instance.OnSuccessfulKnifeHit();

}

if (collision.collider.tag == "Apple")

{

//Destroy(collision.gameObject);

// Destroy(GameObject.Find("Apple"));

//Автоматически унаследует ротацию нового родителя

rb.bodyType = RigidbodyType2D.Kinematic;

transform.SetParent(collision.collider.transform);

//Перемещение в зависимотси от лезвия, которое застряло в бревне

knifeCollider.offset = new Vector2(knifeCollider.offset.x, -0.23f);

knifeCollider.size = new Vector2(knifeCollider.size.x, 0.3f);

//Создание нового ножа

GameController.Instance.OnSuccessfulKnifeHit();

}

//Столкновение с ножом

else if (collision.collider.tag == "Knife")

{

rb.velocity = new Vector2(rb.velocity.x, -2); //Перемещаем быстро в низ

GameController.Instance.StartGameOverSequence(false); //Конец игры

}

}

}

**Листинг 3. Игровой контроллер.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class GameUI : MonoBehaviour

{

[SerializeField]

private GameObject Restart;

[Header("Knife Count Display")] //Заголовок целей организации

[SerializeField]

private GameObject Knives;

[SerializeField]

//Это будет установлено в префабе значка

private GameObject IconKnife;

[SerializeField]

private Color usedKnifeIconColor;

//Делаем активным объект Рестарта

public void ShowRestartButton()

{

Restart.SetActive(true);

}

//Добавление кол-во ножей (дочерних элементов IconKnife в Knives)

public void SetInitialDisplayedKnifeCount(int count)

{

for (int i = 0; i < count; i++)

Instantiate(IconKnife, Knives.transform);

}

private int knifeIconIndexToChange = 0;

//Изменение цвета для брошенного ножа (счётчика ножей)

public void DecrementDisplayedKnifeCount()

{

Knives.transform.GetChild(knifeIconIndexToChange++)

.GetComponent<Image>().color = usedKnifeIconColor;

}

}

**Листинг 4. Управление контроллером.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

//Этот скрипт не сможет работать без GameUI

//Говорим Unity, что это атрибут

[RequireComponent(typeof(GameUI))]

public class GameController : MonoBehaviour

{

//Получаем экземпляр из других скриптов

public static GameController Instance { get; private set; }

[SerializeField]

private int knifeCount;

[Header("Knife Spawning")]

[SerializeField]

private Vector2 knifeSpawnPosition;

[SerializeField]

private GameObject knifeObject; //Это будет сборка ножа.

//Ссылка на GameUI в игровом объекте GameController

public GameUI GameUI { get; private set; }

private void Awake()

{

//Простой вид экземпляра синглтона (мы находимся только в 1 сцене)

Instance = this;

GameUI = GetComponent<GameUI>();

}

private void Start()

{

GameUI.SetInitialDisplayedKnifeCount(knifeCount); //Обновляем интерфейс после запуска игры

SpawnKnife(); //Создаем первый нож

}

//// это будет вызываться из Knife(С# Скрипта)

public void OnSuccessfulKnifeHit()

{

if (knifeCount > 0)

{

SpawnKnife();

}

else

{

StartGameOverSequence(true);

}

}

//Спавн ножа и его параметры

private void SpawnKnife()

{

knifeCount--;

Instantiate(knifeObject, knifeSpawnPosition, Quaternion.identity);

}

//Метод для начала игры

public void StartGameOverSequence(bool win)

{

StartCoroutine("GameOverSequenceCoroutine", win);

}

//Это даёт играку таймер после выиграша

private IEnumerator GameOverSequenceCoroutine(bool win)

{

if (win)

{

yield return new WaitForSecondsRealtime(0.3f);

RestartGame();

}

else

{

GameUI.ShowRestartButton();

}

}

public void RestartGame()

{

//Перезапускаем сцену, перезагружая текущую активную сцену

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex, LoadSceneMode.Single);

}

}

## **Отладка.**

## **Тестирование программы.**

## **Оптимизация программного кода.**

# **Выводы**

Полученные навыки:

* Дизайн уровней Unity3D
* Разработка скриптов в Visual Studio 2019 для игр на базе Unity3D

Полученные умения:

* Работа в Unity 3D
* Работа в Visual Studio 2019

В ходе прохождения практики, были получены новые знания, в среде разработки UNITY и создания игра. В процессе выполнени работы, были освоены компоненты и сторонние сервисы создании игр, от рисованяи текстур до написания технического кода игры.

При прохождение практики было выделенно несколько аспектов для изучения:

* Интерфейс
* Графика
* Тестирование
* Написание технического кода
* Составление ТЗ

# **Дневник практики**

Таблица 1. Дневник практики.

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Содержание работы | Отметка о выполнении  работы | Подпись руководителя практики |
| 26.02 – 31.02 | Обсуждение концепции игры |  |  |
| 2.03 – 13.03 | Работа в Unity3D |  |  |
| 25.03 – 2.04 | Разработка программного содержания. |  |  |
| 12.04 – 15.04 | Работа со скриптами. |  |  |
| 15.04 – 23.04 | Оформление отчёта. |  |  |

# **Список использованной литературы.**

1. «The Python Tutorial» © 2001-2021, Python Software Foundation: https://docs.python.org/3/tutorial/
2. «Как писать игры на Python с помощью Pygame». © PythonRu 2018-2021 — Образовательный блог о Python: <https://pythonru.com/uroki/biblioteka-pygame-chast-1-vvedenie> [электронный ресурс]
3. «Создание игр на Python 3 и Pygame». © 2006 – 2021 «Habr»: <https://habr.com/ru/post/347138/> [электронный ресурс]
4. Документация по библиотеке Pygame языка Python. © 2001-2021. Python Software Foundation: <https://www.pygame.org/docs/> [электронный ресурс]
5. Документация по языку Python. © 2001-2021. Python Software Foundation: <https://www.python.org/doc/> [электронный ресурс]

# **Приложения.**

**Приложение 1.**

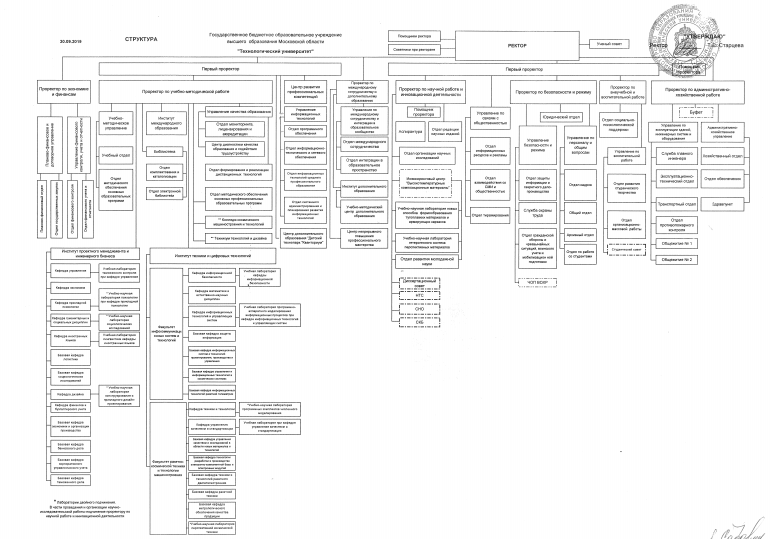


Рис. 4. Организационная структура колледжа