Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Отчёт по:

# ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

Студент группы: 505 В. И. Русских

Руководитель работы: ст. препод. И. А. Шмаков

Барнаул 2022 г.

# РЕФЕРАТ

Полный объём работы составляет 25 страниц, включая 0 рисунков и 0 таблиц.

Во время данной проектной практики была создана компьютерная игра жанра ”платформер” для одного игрока для операционных систем Linux и Windows 10 на языке программирования Python 3.9

Ключевые слова: компьютерная игра, Tkinter, платформер.

Отчёт оформлена с помощью системы компьютерной вёрстки TEX и его расширения XETEX из дистрибутива *TeX Live*.

## СОДЕРЖАНИЕ

Введение . . . . . 4

**1 Глава 1. Используемые программные средства 9**

1.1 Язык программирования . . . . . 9

1.2 Среда разработки . . . . . 9

1.3 Подключаемые библиотеки . . . . . 9

**2 Глава 2. Описание программы 10**

2.1 Методы и классы . . . . . 10

2.2 Алгоритм . . . . . 10

Заключение . . . . . 12

**Список использованной литературы 13**

2.3 Код программы Bunny and Carrot . . . . . 15

# ВВЕДЕНИЕ

В современном мире компьютер играет все большую роль в жизни людей. Так как персональный компьютер со временем стал доступен практически каждому, то он начал использоваться не только для работы, но и для развлечения. Следствием этого является тот факт, что одним из многочисленных способов проведения досуга стали компьютерные игры. Компьютерные игры бывают самых разнообразных жанров: шутеры, головоломки, стратегии, симуляторы, платформеры и многие другие.

В данной технологической практике стоила задача создать игру жанра платформер — жанр компьютерных игр, в которых основу игрового процесса составляют прыжки по платформам, лазанье по лестницам, сбор предметов, необходимых для победы над врагами или завершения уровня.

В ходе создания заданной игры использовалась интегрированння среда разработки PyCharm, так как эта среда обладает достаточно удобным интерфейсом. Языком программирования был выбран Python 3.9, так как он достаточно прост для изучения.

# ПОСТАНОВКА ЗАДАЧИ

Целью практики была разработка программного продукта – простая графическая игру жанра платформер, которой было дано название ”Bunny and Carrot”, в которой объект, в данном случае кролик, должен для завершения игры столкнуться (съесть) с морковкой. Данная игра должна имет окно, которое является игровым полем. С помощью пробела и стрелок вправо и влево кролик перемещается по игровому полю, запрыгивает на платформы и взаимодействует с морковкой. Также игра должна соответствовать некоторым правилам:

1. Игра проходит на поле размером 500 на 700 пикселей.
2. Кролик размещается в самом низу поля, а морковка практически на самом верху.
3. Для победы и завершения игры игрок должен столкнуться с морковкой.
4. Кролик не должен выходить за пределы игрового поля.
5. Кролик не должен ”проваливаться” сквозь платформы.

## ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

1. Наименование программного продукта.
   - 1.1. Компьютерная игра, платформер.
   - 1.2. Условное наименование: КИП.
2. Заказчик и исполнитель работы.
   - 2.1. Заказчик: Кафедра ВТиЭ ИЦТЭФ АлтГУ.
   - 2.2. Исполнитель: студент гр.505 Русских Валентина Игоревна.
3. Цель разработки и функциональное назначение программного продукта.
   - 3.1. Целью разработки является создание компьютерной игры, для развития навыков программирования.
4. Источники разработки.
   - 4.1. Источниками разработки являются:
     - 4.1.1 Формулировка задачи.
     - 4.1.2 Постановка задачи.
     - 4.1.3 Обзор и анализ готового продукта.
5. Системные и технические требования к программному продукту.
   - 5.1. Условия эксплуатации.
     - 5.1.1 Программный продукт должен работать на персональном компьютере стандартной комплектации.
     - 5.1.2 Программный продукт должен запускаться под операционными системами на которых присутствует интерпретатор Python3.
   - 5.2. Требования к используемым библиотекам.
     - 5.2.1 Допускается применение следующих библиотек: Tkinter.
6. Результаты и сроки выполнения работы.
   - 6.1. Разработка технического задания (до 27.09.2021 года).
   - 6.2. Создание проектной документации (до 10.10.2021 года).
   - 6.3. Создание и проектирование КИП (до 07.12.2021 года).
   - 6.4. Сдача продукта в эксплуатацию (до 28.12.2021 года).
7. Стадии и этапы разработки.
   - 7.1. Разработка технического задания.

7.2. Реализация проекта.

7.3. Подготовка проектной документации и пояснительной записки.

7.4. Сдача продукта в эксплуатацию.

8. Порядок сдачи и приемки программного продукта.

8.1. Проверка соответствия продукта ТЗ.

8.2. Тестирование ГР.

8.3. Защита пояснительной записки.

## ЦЕЛИ И ЗАДАЧИ

Для достижения поставленной цели были поставлены следующие задачи:

1. изучить классификацию компьютерных игр и специфику жанра платформер;
2. подготовить виртуальную среду для тестирования программного продукта;
3. в достаточной степени изучить язык программирования Python 3.9
4. изучить графическую библиотеку tkinter;
5. разработать концепцию использования классов;
6. спроектировать программный продукт;
7. написать и протестировать компьютерную игру на нескольких операционных системах.

# 1. ГЛАВА 1. ИСПОЛЬЗУЕМЫЕ ПРОГРАММНЫЕ СРЕДСТВА

## 1.1. Язык программирования

Программа для данной практики была написана на языке Python 3.9, так как уже изначально этот язык программирования был указан в задании, что является вполне разумным решением, так как для выполнения задания требовалось изучить язык программирования в достаточной степени в довольно краткие строки, чему способствует сама простота языка Python 3.9 и просто огромное количество материалов для изучения данного языка. Python поддерживает объектно-ориентированное, структурное, обобщённое, функциональное программирование и метапрограммирование.

## 1.2. Среда разработки

После выбора языка программирования следовала задача выбрать среду разработки. Выбор пал на PyCharm Community Edition, так как в нем удобно работать за счет встроенного автодополнения кода, а также предоставляет понятный и простой интерфейс, который можно настроить.

Сюда же можно добавить графический редактор GIMP, в котором самостоятельно были отрисованы все спрайты, необходимые для дальнейшей работы.

## 1.3. Подключаемые библиотеки

Так как в игре нужно работать с графическими изображениями, то для этого нужна библиотека Tkinter (она встроена в стандартную библиотеку языка), которая как раз отвечает за работу с графическим интерфейсом, что позволяет пользователю взаимодействовать с объектами на экране.

# 2. ГЛАВА 2. ОПИСАНИЕ ПРОГРАММЫ

## 2.1. Методы и классы

Для начала работы с программой необходимо было создать ее графический интерфейс, а именно:

1. Изображение кролика, который будет передвигаться по игровому полю
2. Морковку, после соприкосновения с которой будет считаться, что игрок победил
3. Платформы, на которые должен запрыгивать кролик
4. Изображение фона

Для работы с программой использовались определенные классы:

1. Класс Game - самый главный класс, который управляет остальным кодом
2. Класс Coords - нужен для размещения спрайтов на экране и в объектах которого хранятся позиции всех графических элементов в игре, а также функции для проверки столкновений
3. Класс Sprite - класс-”предок” для всех игровых объектов
4. Класс PlatformSprite - ”потомок” класса Sprite, который принимает аргумент, само изображение платформы, позицию по горизонтали и вертикали, ширину изображения и его высоту
5. Класс BunnySprite - инициализирует и загружает изображение кролика, а также привязывает к нажатию клавиш
6. Класс CarrotSprite - передает изображение морковки и ее координаты

## 2.2. Алгоритм

Начало

1. Создание класса Game
   1.1. создание функции для инициализации игры
   1.2. создание функции для управления игровой анимацией
2. Создание класса Coords
   2.1. создание функции для проверки пересечения по горизонтали
   2.2. создание функции для проверки пересечения по вертикали
3. Создание класса Sprite

3.1. создание функции для пересечения спрайта

3.2. создание функции для возвращения текущей позиции на экране

4. Создание класса PlatformSprite

4.1. создание функции для пересечения спрайта

4.2. создание функции для возвращения текущей позиции на экране

Конец

# ЗАКЛЮЧЕНИЕ

В соответствии с поставленными задачами был создан работоспособный программный продукт - игра-платформер ”Bunny and Carrot”. В игре выполняются все заданные параметры, а в ходе ее разработки была достаточно изучена библиотека Tkinter.

1. Пример ссылки на литературу [4].
2. Пример ссылки на литературу [3].
3. Пример ссылки на литературу [2].
4. Пример ссылки на литературу [1].

## СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ


1. [Электронный ресурс] Бесплатная версия GIMP. — URL: https://gimp.ru.uptodown.com/windows (дата обр. 27.10.2021).
2. [Электронный ресурс] Бесплатная версия PyCharm. — URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обр. 28.10.2021).
3. [Электронный ресурс] Изучаем Python - Марк Лутц. — URL: https://codernet.ru/books/python/izuchaem_python_5-e_izd_tom_1_mark_lutc/ (дата обр. 28.12.2021).
4. [Электронный ресурс] Изучаем программирование на Python - Пол Бэрри. — URL: https://magbook.net/book/36665 (дата обр. 28.12.2021).

## 2.3. Код программы Bunny and Carrot

```
from tkinter import *
import random
import time


class Game:
    def __init__(self):
        self.tk = Tk()
        self.tk.title("Bunny & Carrot")
        self.tk.resizable(0, 0)
        self.tk.wm_attributes("-topmost", 1)
        self.canvas = Canvas(self.tk, width=500,
         ↪  height=700, highlightthickness=0)
        self.canvas.pack()
        self.tk.update()
        self.canvas_height = 700
        self.canvas_width = 500
        self.bg = PhotoImage(file="background.gif")
        w = self.bg.width()
        h = self.bg.height()
        self.canvas.create_image(0, 0, image=self.bg,
         ↪  anchor='nw')
        self.sprites = []
        self.running = True
        self.won = False

    def mainloop(self):
        while True:
            if self.running == True:
                for sprite in self.sprites:
                    sprite.move()
            else:
```

```
                    if self.won == False:
                        self.won = True
                        self.canvas.create_text(250, 350,
                        ↪  fill="white", text="WIN!",
                        ↪  font=("sans-serif", -50))
                        for i in self.sprites:
                            if i.endgame == True:
                                i.end_game()
                self.tk.update_idletasks()
                self.tk.update()
                time.sleep(0.01)
                if self.won:
                    time.sleep(3)
                    quit()


class Coords:
    def __init__(self, x1=0, y1=0, x2=0, y2=0):
        self.x1 = x1
        self.y1 = y1
        self.x2 = x2
        self.y2 = y2


def within_x(co1, co2):
    if (co1.x1 > co2.x1 and co1.x1 < co2.x2)\
            or (co1.x2 > co2.x1 and co1.x2 < co2.x2)\
            or (co2.x1 > co1.x1 and co2.x1 < co1.x2)\
            or (co2.x2 > co1.x1 and co2.x2 < co1.x2):
        return True
    else:
        return False
```

```
def within_y(co1, co2):
    if (co1.y1 > co2.y1 and co1.y1 < co2.y2)\
            or (co1.y2 > co2.y1 and co1.y2 < co2.y2)\
            or (co2.y1 > co1.y1 and co2.y1 < co1.y2)\
            or (co2.y2 > co1.y1 and co2.y2 < co1.y2):
        return True
    else:
        return False


def collided_left(co1, co2):
    if within_y(co1, co2):
        if co1.x1 <= co2.x2 and co1.x1 >= co2.x1:
            return True
    return False


def collided_right(co1, co2):
    if within_y(co1, co2):
        if co1.x2 <= co2.x1 and co1.x2 >= co2.x2:
            return True
    return False


def collided_top(co1, co2):
    if within_x(co1, co2):
        if co1.y1 <= co2.y2 and co1.y1 >= co2.y1:
            return True
    return False


def collided_bottom(y, co1, co2):
    if within_x(co1, co2):
        y_calc = co1.y2 + y
```

```
            if y_calc >= co2.y1 and y_calc <= co2.y2:
                return True
        return False


class Sprite:
    def __init__(self, game):
        self.game = game
        self.endgame = False
        self.coordinates = None

    def move(self):
        pass

    def coords(self):
        return self.coordinates


class PlatformSprite(Sprite):
    def __init__(self, game, photo_image, x, y, width,
     ↪  height):
        Sprite.__init__(self, game)
        self.photo_image = photo_image
        self.image = game.canvas.create_image(x, y,
         ↪  image=self.photo_image, anchor='nw')
        self.coordinates = Coords(x, y, x + width, y +
         ↪  height)
        self.skip_top = True


class BunnySprite(Sprite):
    def __init__(self, game):
        Sprite.__init__(self, game)
        self.images_left = [
```

```
            PhotoImage(file="bunny1left.gif"),
            PhotoImage(file="bunny2left.gif")
        ]
        self.images_right = [
            PhotoImage(file="bunny1right.gif"),
            PhotoImage(file="bunny2right.gif")
        ]
        self.image = game.canvas.create_image(300, 600,
        ↪  image=self.images_left[0], anchor='nw')
        self.x = 0
        self.y = 0
        self.direction = 1
        self.current_image = 0
        self.current_image_add = 1
        self.jump_count = 0
        self.last_time = time.time()
        self.coordinates = Coords()
        game.canvas.bind_all('<KeyPress-Left>',
        ↪  self.turn_left)
        game.canvas.bind_all('<KeyPress-Right>',
        ↪  self.turn_right)
        game.canvas.bind_all('<KeyRelease-Left>',
        ↪  self.stop)
        game.canvas.bind_all('<KeyRelease-Right>',
        ↪  self.stop)
        game.canvas.bind_all('<space>', self.jump)

    def turn_left(self, evt):
        self.x = -2

    def turn_right(self, evt):
        self.x = 2

    def stop(self, evt):
```

```
        self.x = 0

    def jump(self, evt):
        if self.y == 0:
            self.y = -4
            self.jump_count = 0^^I

    def animate(self):
        if self.x != 0 and self.y == 0:
            if time.time() - self.last_time > 0.1:
                self.last_time = time.time()
                self.current_image +=
                 ↪  self.current_image_add
                if self.current_image >= 1:
                    self.current_image_add = -1
                if self.current_image <= 0:
                    self.current_image_add = 1
        if self.x < 0:
            if self.y != 0:
                self.game.canvas.itemconfig(self.image,
                 ↪  image=self.images_left[1])
                self.direction = -1
            else:
                self.game.canvas.itemconfig(self.image,
                 ↪  image=self.images_left[self.current_ima
                self.direction = -1
        if self.x > 0:
            if self.y != 0:
                self.game.canvas.itemconfig(self.image,
                 ↪  image=self.images_right[1])
                self.direction = 1
            else:
                self.game.canvas.itemconfig(self.image,
                 ↪  image=self.images_right[self.current_ima
```

```
                self.direction = 1
        if self.x == 0 and self.y == 0:
            if self.direction > 0:
                self.game.canvas.itemconfig(self.image,
                    ↪ image=self.images_right[0])
            if self.direction < 0:
                self.game.canvas.itemconfig(self.image,
                    ↪ image=self.images_left[0])


    def coords(self):
        xy = self.game.canvas.coords(self.image)
        self.coordinates.x1 = xy[0]
        self.coordinates.y1 = xy[1]
        self.coordinates.x2 = xy[0] + 27
        self.coordinates.y2 = xy[1] + 30
        return self.coordinates

    def move(self):
        self.animate()
        if self.y < 0:
            self.jump_count += 1
            if self.jump_count > 20:
                self.y = 4
        if self.y > 0:
            self.jump_count -= 1

        co = self.coords()
        left = True
        right = True
        top = True
        bottom = True
        falling = True
```

```
if self.y > 0 and co.y2 >=
 ↪  self.game.canvas_height:
    self.y = 0
    bottom = False
elif self.y < 0 and co.y1 <= 0:
    self.y = 0
    top = False

if self.x > 0 and co.x2 >=
 ↪  self.game.canvas_width:
    self.x = 0
    right = False
elif self.x < 0 and co.x1 <= 0:
    self.x = 0
    left = False

for sprite in self.game.sprites:
    if sprite == self:
        continue
    sprite_co = sprite.coords()
    if top and self.y < 0 and collided_top(co,
     ↪  sprite_co) and not sprite.skip_top:
        self.y = -self.y
        top = False
        if sprite.endgame:
            self.game.running = False

    if bottom and self.y > 0 and
     ↪  collided_bottom(self.y, co,
     ↪  sprite_co):
        self.y = sprite_co.y1 - co.y2
        if self.y < 0:
            self.y = 0
        if sprite.endgame:
```

```
                    self.game.running = False
                bottom = False
                top = False

            if bottom and falling and self.y == 0 and
             ↪  co.y2 < self.game.canvas_height and
             ↪  collided_bottom(1, co, sprite_co):
                falling = False
                if sprite.endgame:
                    self.game.running = False

            if left and self.x < 0 and
             ↪  collided_left(co, sprite_co):
                self.x = 0
                left = False
                if sprite.endgame:
                    self.game.running = False

            if right and self.x > 0 and
             ↪  collided_right(co, sprite_co):
                self.x = 0
                right = False
                if sprite.endgame:
                    self.game.running = False

        if falling and bottom and self.y == 0 and co.y2
         ↪  < self.game.canvas_height:
            self.y = 4

        self.game.canvas.move(self.image, self.x,
         ↪  self.y)


class CarrotSprite(Sprite):
```

```
    def __init__(self, game, photo_image, x, y, width,
    ↪ height):
        Sprite.__init__(self, game)
        self.photo_image = photo_image
        self.image = game.canvas.create_image(x, y,
        ↪ image=self.photo_image, anchor='nw')
        self.coordinates = Coords(x, y, x + (width /
        ↪ 2), y + height)
        self.endgame = True

    def end_game(self):
        self.game.canvas.itemconfig(self.image,
        ↪ state='hidden')


g = Game()
platform1 = PlatformSprite(g,
↪ PhotoImage(file="travka.gif"), 0, 652, 500, 48)
platform2 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 300, 610, 100, 10)
platform3 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 150, 550, 100, 10)
platform4 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 150, 250, 100, 10)
platform5 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 150, 150, 100, 10)
platform6 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 150, 350, 100, 10)
platform7 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 150, 450, 100, 10)
platform8 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 300, 500, 100, 10)
platform9 = PlatformSprite(g,
↪ PhotoImage(file="plat3.gif"), 300, 200, 100, 10)
```

```
platform10 = PlatformSprite(g,
 ↪  PhotoImage(file="plat3.gif"), 300, 300, 100, 10)
platform11 = PlatformSprite(g,
 ↪  PhotoImage(file="plat3.gif"), 300, 400, 100, 10)
g.sprites.append(platform1)
g.sprites.append(platform2)
g.sprites.append(platform3)
g.sprites.append(platform4)
g.sprites.append(platform5)
g.sprites.append(platform6)
g.sprites.append(platform7)
g.sprites.append(platform8)
g.sprites.append(platform9)
g.sprites.append(platform10)
g.sprites.append(platform11)

bs = BunnySprite(g)
g.sprites.append(bs)
c = CarrotSprite(g, PhotoImage(file="carrot.gif"),
 ↪  300, 75, 16, 16)
g.sprites.append(c)
g.mainloop()
```