Научно-исследовательская работа

Предмет: Информатика

НАЗВАНИЕ РАБОТЫ: СОЗДАНИЕ ПРОСТЕЙШЕЙ 2D ИГРЫ «ПОЙМАЙ КНОПКУ» НА ЯЗЫКЕ

ПРОГРАММИРОВАНИЯ PYTHON

Выполнил(а):

Саксонов Артем Андреевич

учащийся 5 А класса

МБОУ Школа № 188 Россия, г. Новосибирск

Руководитель:

Масленникова Евгения Владимировна

Учитель иностранных языков

МБОУ Школа № 188 Россия, г. Новосибирск

2023

Введение

В настоящее время процесс информатизации проявляется во всех сферах человеческой деятельности. Использование современных информационных технологий является необходимым условием успешного развития как отдельных отраслей, так и государства в целом.

В современном обществе огромное количество людей проводят свое свободное время за компьютером, работая, играя и зарабатывая деньги или же просто читая новости. Это обычные обыватели: дети, подростки, взрослые, программисты, предприниматели, инженеров и художников. Из всех перечисленных мы бы хотели поговорить о программистах, потому что именно они связаны с нашим проектом. Итак, программист — это специалист, который занимается разработкой исходного кода, алгоритма и компьютерной программы на основе специальных математических моделей. Такой программой может быть операционная система компьютера, видеоигра, web-сайт, мобильное приложение, текстовый редактор и даже алгоритм работы кофемашины. Программный код пишется на специальном языке программирования. Он состоит из обычных слов и некоторых специальных символов, присвоенных своему языку. В наше время насчитывается более 100 языков программирования, но самые распространенные из них – Java, Python, PHP, C#, JavaScript, C, C++, Objective-C, Swift, ABC Pascal и другие. Так как создание нашей игры будет происходить на языке программирования Python, расскажем о нём поподробнее. Python — это высокоуровневый язык программирования, который используется в различных сферах IT, таких как машинное обучение, разработка приложений, web и другие. Python очень популярен среди новичков и любителей программирования, потому что он очень прост и удобен в использовании. В 2019 году Python стал самым популярным языком программирования, обогнав Java. От выбора языка, программист зависит от конкретной задачи и собственных знаний того или иного языка. Его выбор полностью зависит от его намерений, например,

создать игру, мобильное приложение или же собственный сервер. Квалифицированный программист может уверенно использовать 2-4 языка. У программистов широкая сфера приложения профессиональных знаний, к такой сфере можно отнести IT-сферу. Сокращение IT означает Information Technology, что в переводе с английского – информационные технологии. IT-сфера – одна из самых перспективных отраслей в мировой экономике, это деятельность по изучению и разработке процессов сбора, хранения, поиска, обработки и передачи информации, а также изучение и разработка процессов управления информацией с помощью компьютеров и других устройств. Сегодня IT-специалистами называют относительно большую группу людей профессионалов, занимающихся одним и тем же видом деятельности и имеющих общие интересы и задачи, чья сфера деятельности так или иначе связана с информационными технологиями в целом и разработкой программного обеспечения. Сфера IT и программирования с каждым днем набирает обороты и все больше и больше развивается в новых технологиях по всему миру вперед, упрощая жизнь обычных людей.

Наш проект направлен на создание 2D-игры на языке программирования Python, в последствии чего, мы будем знакомиться с языком на более высоком уровне, изучая его функции, классы, структуры, библиотеки, циклы, операторы и многое другое, а также найдем новую информацию о строении, структуре и особенностях компьютерных 2D-игр. **Целью** нашей работы является создание своей 2D-игры на языке программирования Python. Основной **задачей** ставлю перед собой изучить библиотеку «tkinter» для создания алгоритма игры. В самом начале нашей работы нужно определиться с редактором кода, библиотеками и классами, которые будут использоваться нами при написании игры. Мы выбирали среди огромного количества библиотек и классов, имеющихся в Python, которые в последствии увеличат возможности в моей игре. С выбором редактора кода у нас не было никаких проблем, ведь за основу всех проектов, созданных нами, мы использовали Python . В момент создания

нашей игры мы планируем еще больше углубиться в изучение языка программирования Python, выйдя на более высокий уровень знаний, а также обратить наше внимание на технологии создания игры, конечно, не без помощи интернета. После того, как игра будет готова я хочу показать ее своим друзьям и окружающим меня людям, послушать их мнение об этой игре и поделиться с ними своими впечатлениями во время разработки и написания кода для этой игры.

ГЛАВА 1. ИСТОРИЯ ЯЗЫКА PYTHON И СТРУКТУРА КОМПЬЮТЕРНЫХ ИГР

1.1 История языка программирования Python.

Python иногда используется для написания десктопных приложений и, конечно, тотально доминирует в сфере машинного обучения. Дескоптные приложения -это программы, обрабатываемые на стороне клиента и запускаемые в виде обыкновенного исполняемого файла на устройстве пользователя. Кроме того, на нём создаётся много прототипов, которые позволяют быстро набросать структуру, функционал и внешний вид будущего проекта. Создатель языка Python — нидерландский программист Гвидо ван Россум.

Он был участником проекта по написанию ABC Pascal, языка для обучения программированию. В конце 1989 года Гвидо приступил к разработке своего

нового языка и задумал его как потомка ABC, способного к обработке исключений и взаимодействию с операционной системой Amoeba. На тот

момент он работал в центре математики и информатике в Нидерландах.

Он назвал его в честь британского комедийного шоу «Monty Python», что в

переводе означает «Летающий цирк Монти Пайтона», которое было популярно в начале 1970-х годов. Это телешоу позволяло автору расслабиться и отвлечься от разработки языка. Однако, несмотря на настоящее происхождение названия, для людей более очевидно связывать Python со словом «змея». Этому также способствует логотип, на котором изображена рептилия. И хотя создатель языка не раз говорил, что название никак не связано со змеями, повлиять на мнение общества так и не удалось. Так и получился Python.

1.2 Развитие Python

Python свободно распространялся через интернет и со временем у него появились последователи — люди, заинтересованные в развитии этого языка программирования. Первая публикация Python состоялась в феврале 1991 года — это была версия 0.9.0. В 1994 году Гвидо опубликовал Python 1.0, а потом одна за другой вышли и другие версии: до 2.0 язык обновился в октябре 2000, до 3.0 — в декабре 2008. В октябре 2021 мир увидела самая свежая версия — Python 3.10.0. Синтаксис Питона всегда выделял его на фоне других языков программирования. Он не страдает избыточностью, схожесть синтаксиса с обычным английским позволяет понять код даже обычному пользователю, кроме того, программист пишет меньше строк кода, потому что нет необходимости использовать символы: «;», «{», «}». Вложенность обозначается отступами, что повышает читаемость кода и приучает новичков к правильному оформлению.

Простота отчасти обусловлена тем, что Питон написан на основе языка ABC, который использовался для обучения программированию и повседневной работы людей, не являющихся программистами.

Python упрощает написание кода и делает разработку быстрой, всё потому что он обладает следующими особенностями:

- Динамическая типизация. Программисту не нужно указывать тип переменных, язык присвоит его сам. Константы, переменные и числовые значения разных типов, участвующие в одной операции, автоматически приводится к нужному по определённым правилам.

- Автоматическое выделение памяти. Программисту не нужно самостоятельно выделять память под что-либо. С одной стороны, это уменьшает контроль программиста над программой, а с другой, разработка значительно ускоряется.

- Удобный возврат нескольких значений функцией. Их можно перечислить через запятую, и они автоматически преобразуются в список, то есть не нужно выделять память и передавать указатели в функцию.
- Сборщик мусора. Если объект становится бесполезным или на него перестаёт что-либо ссылаться, он автоматически удаляется сборщиком мусора. Сборщик мусора позволяет оптимизировано использовать память и не удалять бесполезные объекты вручную.
- Привязка типа данных. Тип данных привязан к значению, а не к переменной. То есть значение - это какой-то объект с атрибутами, которые определяют его тип и другие характеристики, а переменная - просто ссылка на этот объект. Такой подход позволил обойтись без явного определения типов и значительно упростил повторное присваивание значения переменной.
- a, b = b, a. Эта строка меняет местами значения переменных, теперь то, что было в a, находится в b и наоборот. Такое возможно, потому что Питон сначала рассматривает переменные справа от знака «=» и помещает их в список, то же он делает с элементами слева от «=», затем он связывает каждый элемент правого списка с левым. Таким способом можно обменивать значения не только двух переменных, но и трёх, пяти и так далее.
- Интерпретируемый язык. Написанный код не нужно компилировать, достаточно запустить его и получить результат. Более того, можно работать в интерактивном режиме и получать результат буквально после каждой операции.

Python сочетает в себе и простоту и мощный инструментарий. Его можно использовать для создания прототипа практически любой программы.

Чтобы ускорить разработку, часть программы, обычно не сильно влияющую

на скорость работы пишут на Питоне.

Именно благодаря простоте этот язык программирования смог занять доминирующее место в сфере машинного обучения. Люди, так или иначе связанные с наукой, предпочитают не тратить много времени на такие вещи, как написание кода, поэтому Python отлично подошёл для реализации поставленных перед ними задач.

1.3 Актуальность использования языка программирования Python

Несмотря на то что языку уже более 31 года, он популярен среди программистов всего мира. Python используется почти в каждом среднем или крупном проекте, если не как основной инструмент разработки, то как инструмент для создания прототипа или написания какой-то его части. Он собрал вокруг себя огромное сообщество разработчиков, предпочитающих именно этот язык программирования

Преимущества, чем хорош Python

Специалисты в этой сфере выделяют огромное множество преимуществ у Python. К ним относятся:

- Простота и минималистичность. Чтение программы на Python очень напоминает текст английского языка, хотя и достаточно строгого. Такая конструкция Python является одной из его самых сильных сторон.
- Простота синтаксиса. Низкий порог0вхождения (при изучении нового языка программирования можно быстро и без больших усилий добиться значительных результатов). Код языка чистый и понятный, без лишних символов и выражений.
- Интерпретируемость. Интерпретатор Python есть для всех популярных платформ и по умолчанию входит в большинство дистрибутивов Linux.

Программа просто выполняется из исходного текста. Python сам преобразует этот исходный текст в некоторую промежуточную форму, называемую байт-кодом. Байт-код -

это код виртуального процессора, который обычно интерпретируется в машинный. Всё это заметно облегчает использование Python, поскольку нет необходимости заботиться о

- Расширяемость и гибкость. Python можно легко расширить для взаимодействия с другими программными системами и библиотеками или встроить в другие программы в качестве компонента. Например, если вам нужно, чтобы какая-то часть программы работала очень быстро или вы вынуждены скрыть часть алгоритма, вы можете написать эту часть программы на C или C++, а затем вызывать её из программы на Python.

- Кроссплатформенность. Python имеет возможность запускать программы не только для программных компьютеров, но и для телефонов, приставках и другое. Все ваши программы смогут запускаться на любой из этих платформ без каких-либо изменений, если только вы не использовали системно-зависимые функции.

- Стандартизированность. У Python есть единый стандарт для написания кода — Python Enhancement Proposal или PEP, что в переводе означает Предложения по развитию Python, благодаря чему язык остаётся читабельным даже при переходе от одного программиста

к другому.

- Open Source. Open Source – это обширный и постоянно растущий ресурс для разработчиков, изучая который, они прокачивают свои навыки, начинают понимать больше и становятся лучше как программисты. У интерпретатора Python открытый код, то есть любой, кто заинтересован в развитии языка, может поучаствовать в его разработке и улучшении.

- Большое сообщество. Вокруг Python образовалось дружественное сообщество, которое поможет новичку или уже опытному разработчику

и разобраться в его проблеме. Во всём мире проходит много мероприятий, где можно познакомиться с коллегами и узнать много нового.

- Востребованность на рынке труда и поддержка гигантами IT-сферы. Python-разработчики0востребованы во многих проектах и им несложно найти работу. Разработку на Python ведут в Google, Facebook, Spotify, Netflix, Microsoft Intel, а в России — «Яндекс», «ВКонтакте» и «Сбербанк». Это серьёзно влияет а статус языка.

Недостатки Python

У Python, как и у любого другого языка, есть не только плюсы, но и минусы,

а среди разработчиков - не только фанаты, но и ненавистники. К недостаткам

Python можно отнести следующее:

- Низкая производительность. Python требует высоких вычислительных мощностей серверов и компьютеров. Это делает его не таким быстрым, он отстаёт от других языков по производительности. В условиях сильного развития мощности это не так заметно, как раньше, но всё равно даёт о себе знать. Чтобы

нивелировать эту проблему, разработчики обращаются к языку программирования C для реализации проблемного участка кода. -Динамическая типизация. Python относится к языкам с динамической типизацией, что делает его0невероятно гибким при разработке.

Но несмотря на это, он потребляет0много ресурсов и имеет низкую скорость выполнения программы.

- Глобальная 0 блокировка интерпретатора (GIL). Это способ синхронизации потоков,0 который используется в некоторых интерпретируемых языках программирования. Хотя GIL является самым простым способом избежать конфликтов при одновременном обращении

разных потоков к одним и тем же участкам памяти, у такого подхода есть недостаток — ограничение параллельности вычислений. Также он не позволяет достигать высокой эффективности вычислений при работе на многоядерных системах.

Что можно написать на Python

Python используют во многих областях программирования, поэтому на нём можно создать что душе угодно:

- Интерфейс и внутренний функционал web-сайта. Программист легко может работать со связями URL адресов, обращениями к базам данных и созданием HTML файлов, которые пользователь видит в браузере.

- Blockchain. Блокчейн — это последовательная цепочка блоков, где каждый блок содержит информацию и всегда связан с предыдущим. Технология особенно популярна в финансовой сфере и криптовалюте.

Блокчейн совмещает в себе защищенность и открытость информации, он позволяет получить доступ к данным из любой точки мира, но в тоже время его практически невозможно взломать, данные хранятся на каком-то главном компьютере, а взламывать каждый блок очень затратно и долго.

- Бот. Бот - Это программа, автоматически выполняющая какие-либо действия в заданное время или в ответ на поступивший сигнал. Боты могут примитивно симулировать поведение человека, поэтому они часто используются для работы в технической поддержке и поиска информации в интернете

-База данных. База данных — это информация, систематизированная по общим признакам и специальным правилам. В любом большом проекте используются базы данных, в них хранится информацию о пользователях, изменениях в программе и т. д.

- VR-объекты. VR дополняет физический мир с помощью виртуальных технологий. То есть виртуальные объекты проецируются на реальное окружение, и имитируют признаки и поведение обычных физических

объектов.
- Игра. В Python в основном создаются самые простые игры, начиная от змейки закачивая 2D платформером. Создавать 3D игры – очень и очень трудно. Python используется либо для разработки прототипа, либо для реализации какой-то части игры. Для написания простенькой игры можно воспользоваться библиотекой Pygame или Arcade, которые дают все необходимые инструменты и возможности для создания небольшой 2D игры.
- Язык программирования. Python достаточно высокоуровневый язык, поэтому создавать на нём ещё один язык программирования будет глупо, но возможно. Полезнее будет разработать интерпретатор или библиотеку для Python или другого языка программирования.
Конечно, это не весь перечень вещей, которые поддаются созданию на Python. Создать можно что угодно, начиная с обычного калькулятора, заканчивая масштабными работающими серверами, необходимо только время и знания.

ЦПРАКТИЧЕСКАЯ ЧАСТЬ

ГЛАВА 2. СОЗДАНИЕ СВОЕЙ 2D ИГРЫ

Приступаем к нашей работе. Занимаясь на курсах программирования, я смог записать программу через сайт. Первыми шагами к написанию нашей работы было ознакомление со всей нужной нам информацией, а именно с внутренней составляющей компьютерных игр и языком программирования Python. После чего мы начали прописывать определенные части работы, чтобы в последствии упростило нам написание нашего кода. Когда мы были готовы приступать к нашей работе, мы открыли свой редактор кода, который используем для написания программ на Python, который прост в чтении языка, имеет приятное и понятное для глаз оформление и удобен в устранении своих ошибок при написании кода программы. Далее мы создали файл своей работы, чтобы производить дальнейшие сохранения по ходу написания кода.

Написание нашей игры начиналось с выбора библиотек, классов и модулей. У Python существует 2 библиотеки, которые подходят для создания игр: Pygame и Arcade. Выбрал один из подвидов библиотек «tkinter».(Приложение 1)

Создание главного окна

Объект окна верхнего уровня создается при обращении к классу Tk модуля tkinter. Переменную, связанную с объектом-окном принято называть root (но можно назвать и другим именем). Рассмотрим пример создания главного окна:

```
from tkinter import *

root = Tk() # создаем переменную

root.title("Заголовок окна программы") #заголовок окна

root.geometry("400x200") # начальные размеры окна
```

input()

Если не установить геометрию окна, то размер окна будет подогнан под размер виджетов, содержащихся в окне.

Работа с виджетами

В библиотеки tkinter существует множества виджетов. Для примера рассмотрим виджет «Button».

У класса Button есть обязательный параметр — объект, которому кнопка принадлежит (у нас это окно root). Полный список виджетов мы рассмотрим в следующих статьях. Пример создания кнопки:

but1 = Button(root)

У кнопки есть много свойств: надпись, цвет фона, размер и т.д. Например, добавим текст нашей кнопке:

but1["text"] = "Название кнопки"

Чтобы наша кнопка появилась в окне программы необходимо использовать метод pack(). Добавим в наше окно кнопку:

```
from tkinter import *

root = Tk()

root.title("Заголовок окна программы")

root.geometry("400x200")

but1 = Button(root)

but1["text"] = "Название кнопки"
```

```
but1.pack()

input()
```

Если вы заметили, то при нажатии на кнопку ничего не происходит т.к. у нас нету события. Действия(алгоритм) часто оформляют в виде функции, а затем вызывают необходимым событием. Создадим функцию:

```
def hello(event):

  print("Привет, программист!")
```

Параметр event — это какое-либо событие. Событие нажатия левой кнопки мыши выглядит так: <Button-1>. Требуется связать это событие с обработчиком (функцией hello). Для связи предназначен метод bind. Синтаксис:

```
but1.bind("<Button-1>",hello)
```

Для закрепления теоретического материала соберем все наши примеры в одну программу. Листинг:

```
from tkinter import *

def hello(event):

  print("Привет, программист!")

root = Tk()

root.title("Заголовок окна программы")

root.geometry("400x200")

but1 = Button(root)

but1["text"] = "Название кнопки"
```

```
but1.bind("<Button-1>",hello)

but1.pack()

root.mainloop()

input()
```

Если запустить данную программу и нажать на кнопку, то в консоли появится строчка «Привет, программист!».

Строчка root.mainloop() должна быть всегда в конце скрипта. Окно не появится, пока мы не вызовим метод mainloop().

Ниже прописываю поэтапное создание игры(Приложение 2)

```
# Игра "Поймай кнопку"
# Игра длится 60 секунд. За каждое успешное чекание кнопки начисляются очки
# Перескакивание кнопки происходит с ускорением 5 мс в секунду. Начальная скорость 1 секунда
# Кнопка на самом деле является псевдокнопкой - виджетом Label, на который навешано прерывание по нажатию левой клавиши мыши
# Основой алгоритма игры являются два прерывания (параллельные потоки):
# 1. Прерывание раз в секунду для отсчёта игрового времени и ускорения перескакивания кнопки
# 2. Ускоряющееся прерывание для перебрасывания кнопки в другое, случайное место на игровом поле
# В игре используются следующие глобальные переменные, виджеты и их обозначения:
# w - размер экрана пользователя по ширине; h - размер экрана пользователя по высоте
# i - текущее значение времени; j - текущее значение скорости перескакивания кнопки
# flag - флаг (маркер) начала игры. True - игра идёт, False - игра остановлена/не идёт/режим "меню"
# l1 - надпись "Очки", l2 - надпись "Время", l3 - сервисная надпись (текст инструкции или финального сообщения)
# x - виджет "Очки", y - виджет "Время", btn - виджет псевдокнопки

from tkinter import *    # Импортируем графическую библиотеку TKinter
```

```
import random           # Импортируем библиотеку работы со случайными числами
import threading        # Импортируем библиотеку работы с прерываниями
import time             # Импортируем библиотеку работы с системным временем
from PIL import ImageTk, Image    # Импортируем библиотеку работы с изображениями

window = Tk()           # Создание окна приложения и присваивание его переменной window
window.title("Поймай кнопку")     # Наше приложение называем "Поймай кнопку"
w=window.winfo_screenwidth()//2; h=window.winfo_screenheight()//2  # Считываем размеры экрана монитора пользователя и делим эти размеры пополам
window.geometry(f"{w}x{h}+{window.winfo_screenwidth()//2-w//2}+{window.winfo_screenheight()//2-h//2}")  # И делаем игровое поле в половину экрана
window.resizable(False, False)    # Запрещаем пользователю изменять размеры экрана

path = 'phon.jpeg'              # Прописываем путь к фоновому рисунку
image = Image.open(path)        # Заносим рисунок в переменную image
image = ImageTk.PhotoImage(image) # Объявляем изображение частью игрового поля
panel = Label(image=image)      # Всё игровое поле будет виджетом Label с фоновым рисунком
panel.place(x=0,y=0)            # Размещаем фоновое изображением по адресу x=0, y=0 (левый верхний угол)

# Текст инструкции
txt1="""В этой игре требуется поймать кнопку левой клавишей мыши.
Игра длится ровно 1 минуту.
Чем быстрее прыгает кнопка,
тем больше очков начислятся за каждую успешную охоту.
Попробуйте набрать 100 очков! Удачи!"""

# Прерывание, которое выполняется раз в секунду для хода времени
def time():
    global i,j,flag      # Значение текущего времени i, ускорения j и флаг начала игры flag делаем глобальными
    i=int(y.get())-1     # Считываем текущее значение времени, декрементируем его
    j=j-0.005            # Найденная в ходе экспериментов константа 5 мс, на которую ускояется игра каждую секунду
    if i:                # Проверяем, истекла ли игровая минута?
```

```
        y.delete(0,END)   # Время ещё не закончилось, очищаем поле времени
        y.insert(0,i)     # Записываем в поле времени новое значение времени
        threading.Timer(1, time).start()   # Запускаем прерывание-ход времени на 1 секунду
        return            # Возвращаемся из подпрограммы time в этой ветке
    flag=False            # Если время истекло, очищаем флаг начала игры
    if int(x.get())<100:  # Проверяем, больше или меньше 100 очков набрал игрок и в зависимости от этого готовим разный текст в переменной txt2
        txt2="Плохо! Вы набрали всего {} очков. Попробуйте ещё"
    else:
        txt2="Здорово! Вы набрали целых {} очков. Ставьте новые рекорды!"
    txt2=txt2.format(x.get())    # Внутрь подготовленной предыдущей фразы вставляем количество набранных очков
    x.delete(0,END); y.delete(0,END)   # Очищаем поля времени и результатов
    l1.place_forget(); l2.place_forget(); x.place_forget(); y.place_forget(); btn.place_forget()  # Скрываем виджеты времени, результата и кнопку
    l3.config(text=txt2)         # Меняем в виджете l3 текст с инструкции на финальный текст
    l3.place(x=0,y=0)            # Размещаем виджет l3 по адресу x=0, y=0 (левый верхний угол)

# Прерывание, которое выполняется с ускорением (5 мс за секунду) для установки кнопки в новом месте
def interrupt():
    global w,h,j,flag            # Размеры экрана, ускорение и флаг игры берём снаружи (это глобальные переменные)
    if not(flag): return         # Если игра уже закончилась, выходим
    btn.place(x=random.randrange(w-8*w//200),y=random.randrange(40, h-16*h//300)) # Ставим кнопку по случайным координатам в пределах игрового поля
    btn.config(bg="blue")        # Кнопку делаем синей (вдруг на предыдущем ходе её зачекали и она уже красная)
    window.update()              # Перерисоываем игровое поле с учётом нового положения кнопки
    threading.Timer(j, interrupt).start()   # Запускаем прерывание-следующее исчезновение кнопки с учётом ускоряющегося времени

# Подпрограмма обработки нажатия кнопки
def clicked(event):
    if btn.cget("bg")=="red": return    # Если мы уже зачекали кнопку (красная), пока она не перескочила в новое положение, повторные нажатия недопустимы
    btn.config(bg="red")         # Если мы только-только зачекали кнопку, делаем её красной
    global i                     # Для динамического начисления очков текущее значение времени берём снаружи (глобальная переменная)
```

```
    n=6-int(str(i).zfill(2)[0])  # Текущее значение времени переводим в текст, нормируем до двух разрядов, выделяем левую цифру (десятки секунд от минуты)
    n=int(x.get()) + n          # Считываем текущее значение счётчика очков, инкременируем его + добавляем премию за скорость
    x.delete(0,END)             # Очищаем поле результатов
    x.insert(0,n)               # Записываем новое значение счётчика очков

# Прерывание, которое выполняется для начала игры
def begin():
    global flag                 # Переменную начала игры flag делаем глобальной, доступной для всех других прерываний
    flag=True                   # Устанавливаем флаг начала игры flag
    l3.place_forget()           # На всякий случай скрываем инструкцию или финишный текст от предыдущей игры (вдруг это не первая игра?)
    l1.place(x=0,y=0)           # Размещаем виджет - надпись "Очки"
    l2.place(x=80,y=0)          # Размещаем виджет - надпись "Время"
    x.place(x=40,y=2)           # Размещаем виджет - поле результатов по адресу x=40, y=2
    x.insert(0,0)               # В поле результатов записываем ноль
    y.place(x=125,y=2)          # Размещаем виджет - поле времени по адресу x=125, y=2
    y.insert(0,60)              # В поле времени записываем 60 (продолжительность игры)
    threading.Timer(1, time).start()   # Создаём прерывание в 1 секунду для запуска отсчёта времени
    global j; j=1               # Переменную j делаем глобальной. Начальная скорость перескакивания кнопки 1 секунда
    threading.Timer(j, interrupt).start()   # Создаём прерывание-ускорение для кнопки (начальная скорость 1 секунда)

# Прерывание, которое выполняется для вывода инструкции
def instruction():
    global flag                 # Значение переменной flag берём снаружи
    if flag: return             # Если игра уже запущена, то кнопка инструкции заблокирована
    l3.config(text=txt1)        # Если игра не идёт, в виджет l3 прописываем текст инструкции
    l3.place(x=0,y=0)           # Выводим текст инструкции по адресу x=0, y=0

# Прерывание, которое выполняется для выхода из игры
def finish():
    window.destroy()            # При нажатии кнопки "Выход" уничтожаем (закрываем) окно приложения
```

```
# Формируем все виджеты
menu = Menu()                    # Создаём виджет "Меню"
menu.add_command(label="Играть", command=begin)          # К виджету "Меню" добавляем пункт "Играть" и привязываем к нему подпрограмму begin
menu.add_command(label="Инструкция", command=instruction) # К виджету "Меню" добавляем пункт "Инструкция" и привязываем к нему подпрограмму instruction
menu.add_command(label="Выйти", command=finish)          # К виджету "Меню" добавляем пункт "Выйти" и привязываем к нему подпрограмму finish

l1=Label(text="Очки: ")            # Формируем виджет-надпись "Очки"
x=Entry(width=5, justify="right")   # Формируем поле очков
l2=Label(text="Время: ")           # Формируем виджет-надпись "Время"
y=Entry(width=7, justify="right")   # Формируем поле времени
l3=Label(font=("Helvetica", "14", "italic"))   # Формируем виджет сервисных сообщений (инструкций и финальных сообщений). Вначале оно пустое

btn=Label(width=w//200, height=h//300, bd=2, bg="blue")   # Создаём псевдокнопку в виде виджета Label и задаём её размеры
btn.bind('<Button-1>', clicked)                          # К этой псевдокнопке привязываем событие - нажатие левой клавиши мыши (выполняется подпрограмма clicked)

window.config(menu=menu)           # Запускаем игру - выводим меню
flag=False                         # Устанавливаем флаг, что игра ещё не началась

window.mainloop()                  # Зацикливаем игру - ожидаем действий пользователя, пока нет действий - процессор спит
```

Инструкция как скачать игру.

https://drive.google.com/file/d/1eCDFWjvTn1KgTj-3XYVEap9BnP7UKd1r/view?usp=sharing

Для того, чтобы запустить игру на компьютере нужно по ссылке скачать приложение.

https://drive.google.com/file/d/1sPBK8ShmUoO0EeJjinlyKCKtEgSjpDBa/view?usp=sharing

ЗАКЛЮЧЕНИЕ

В ходе моей проделанной исследовательской работы нам удалось добиться желаемого результата. У нас получилось создать игру на языке программирования Python 3.11, конечно, не без каких-либо проблем и недочётов. К такому недочёту можно отнести отсутствие звуков, что в дальнейшем мы постараемся исправить. Цель и задача были достигнуты в ходе работы. Мы ожидали, что создание игры не будет какой-то тяжелой работой, но оказалось все совсем иначе. мы справились с нашими задачами и поставленной целью и в дальнейшем постараемся доработать свою игру и выйти на еще более высокий уровень. Я попросил своих друзей оценить мою игру и что удивительно, игра, по их мнению, оказалась очень даже хороша для первого раза и понравилась всем.

Список литературы

1. Арсак Ж. Программирование игр и головоломок / Ж. Арсак. – Наука, 1990. – 76с.

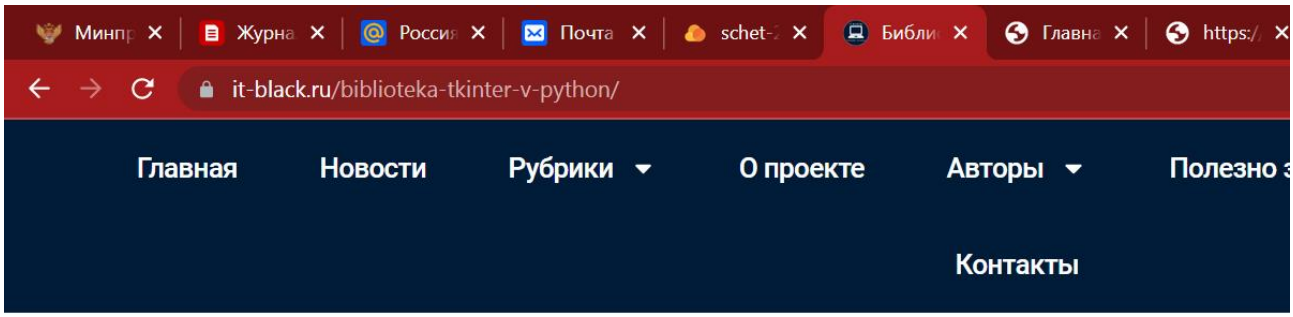
2. Википедия: Python. – Режим доступа: https://ru.wikipedia.org

3. Tech: Python – краткий обзор языка и его назначения. – Режим доступа: https://techrocks.ru

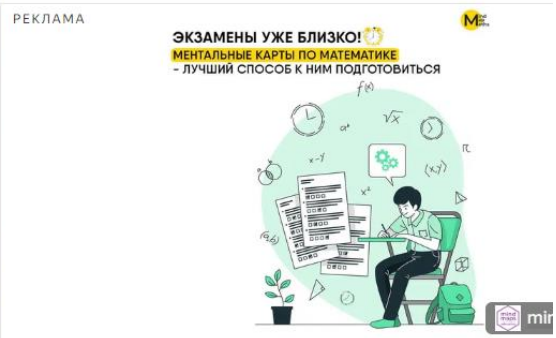
4. Трофимов В. Программирования игр, создание с нуля / В. Трофимов. – 2020. [Электронная книга]

5. Яндекс Дзен: Дзен питона. – Режим доступа: https://zen.yandex.ru

ПРИЛОЖЕНИЕ 1



Библиотека Tkinter в



ПРИЛОЖЕНИЕ 2

artem npk.py - C:\Users\8\Desktop\artem npk.py (3.11.1)

File Edit Format Run Options Window Help

```
from tkinter import *
import random
import threading
import time

window = Tk()    # Создание
window.title("Поймай кнопку")  # Название
w=600; h=600                   # Задаём желаемые размеры экрана 600x600 и устаналиваем окно по центр
window.geometry(f"{w}x{h}+{window.winfo_screenwidth()//2-w//2}+{window.winfo_screenheight()//2-h//2}

# Прерывание, которое выполняется раз в секунду для хода времени
def time():
    global i          # Значение текущего времени делаем глобальным
    i=int(y.get())-1  # Считываем текущее значение времени, декременируем его
    global j          # Значение ускорения делаем глобальным
    j=j-0.005         # С каждым заходом ускоряемся
    if not(i):
        window.destroy()
    y.delete(0,END)   # Очищаем поле времени
    y.insert(0,i)     # Записываем новое значение времени
    threading.Timer(1, time).start()

# Прерывание, которое выполняется со всё ускоряющимся значением (устанавливаем кнопку в другом месте
def interrupt():
    btn.place(x=random.randrange(575),y=random.randrange(60, 575)) # Ставим кнопку по случайным коор
    btn.config(bg="blue")
    window.update()
    global j
    threading.Timer(j, interrupt).start()

# Подпрограмма обработки нажатия кнопки
def clicked(event):
    if btn.cget("bg")=="red": return    # Если мы уже зачекали кнопку, пока она не исчезла, повторны
    btn.config(bg="red")
    global i                    # Считываем текущее значение времени
    n=6-int(str(i).zfill(2)[0])  # Сохраняем в другую переменную, нормируем и выделяем левую часть
    n=int(x.get()) + n           # Считываем текущее значение счётчика, инкременируем его + добавляе
    x.delete(0,END)              # Очищаем поле счётчика
```

artem npk.py - C:\Users\8\Desktop\artem npk.py (3.11.1)

File Edit Format Run Options Window Help

```
# Прерывание, которое выполняется со всё ускоряющимся значением (устанавливаем кнопку в другом месте
def interrupt():
    btn.place(x=random.randrange(575),y=random.randrange(60, 575)) # Ставим кнопку по случайным коор
    btn.config(bg="blue")
    window.update()
    global j
    threading.Timer(j, interrupt).start()

# Подпрограмма обработки нажатия кнопки
def clicked(event):
    if btn.cget("bg")=="red": return    # Если мы уже зачекали кнопку, пока она не исчезла, повторны
    btn.config(bg="red")
    global i                        # Считываем текущее значение времени
    n=6-int(str(i).zfill(2)[0])     # Сохраняем в другую переменную, нормируем и выделяем левую часть
    n=int(x.get()) + n              # Считываем текущее значение счётчика, инкременируем его + добавляе
    x.delete(0,END)                 # Очищаем поле счётчика
    x.insert(0,n)                   # Записываем новое значение счётчика

# Формируем поле подсчёта количества очков
Label(text="Очки: ").place(x=0,y=0)
x=Entry(width=5, justify="right")
x.place(x=40,y=2)
x.insert(0,0)

# Формируем поле времени
Label(text="Время: ").place(x=80,y=0)
y=Entry(width=7, justify="right")
y.place(x=125,y=2)
y.insert(0,60)   # Начальное значение времени

# Создаём псевдокнопку
btn=Label(width=4, height=2, bd=2, bg="blue")
btn.bind('<Button-1>', clicked)

# Создаём прерывание в 1 секунду для отсчёта времени
threading.Timer(1, time).start()
# Создаём прерывание-ускорение для кнопки. Начальная скорость 1 секунда
j=1
```