**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное**

**учреждение Московской области**

**«Люберецкий техникум имени Героя Советского Союза,**

**лётчика-космонавта Ю.А. Гагарина»**

|  |
| --- |
|  |
|  |
|  |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(ДИПЛОМНАЯ РАБОТА)**

На тему: Исследование процесса автоматизации системы делопроизводства в Военном комиссариате городов Люберцы, Дзержинский, Котельники и Лыткарино Московской области

по специальности: 46.02.01 Документационное обеспечение управления и архивоведение

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнила |  | Д.О. Фоменко |  |
| Руководитель |  | Л.А. Акулина |  |

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора по информационным технологиям

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.В.Капранова /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022г.

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное**

**учреждение Московской области**

**«Люберецкий техникум имени Героя Советского Союза,**

**лётчика-космонавта Ю.А. Гагарина»**

|  |  |  |
| --- | --- | --- |
| **РАССМОТРЕНО**  Предметно-цикловой комиссией  технологического профиля  Протокол № \_\_ от «\_\_» \_\_\_\_\_\_\_\_\_\_\_2022 г.  Председатель МЦК\_\_\_\_\_\_\_\_\_Р.С.Макаревич |  | **УТВЕРЖДАЮ**  Первый заместитель директора  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Корнеев  https://lh4.googleusercontent.com/hM489EcerarBwA_W6IHpr56qibD5yOZD6BfqS6n64JMQUOuXbe0v3q7GnPiNH6Aru_GaD3rwczQiJSZVMEmnXqdmh4-tjPKlXPfosNa8wpxyWr17CHyC4xx3_xafrRdKkYo6dZhZNA0KQ0pycQ  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**(ДИПЛОМНАЯ РАБОТА)**

Студент(ка) \_\_\_\_\_\_\_\_\_\_\_Фоменко Данила Олегович\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О)

Группа № 185  специальность 09.02.07 Информационные системы и программирование \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема работы «Создание игры на объектно-ориентированном языке Python»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов по индивидуальной теме):

1 Выполнить обзор темы компьютерных игр

2 Провести исследование предметной области

3 Проанализировать существующие среды разработки для создания компьютерной игры.

4 Разработать структуру

5 Разработать игру

6 Провести тестирование

Руководитель ВКР \_\_\_Жирнова Юлия Витальевна\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Задание получил \_\_ Фоменко Данила Олегович\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

«\_\_*05*\_\_» \_\_\_\_\_*апреля*\_\_\_\_\_ 20*22*г.

Срок предоставления работы «\_*10*\_» \_\_\_*июня*\_\_\_ 20*22*г.

**Содержание**

[**Введение** 3](#_Toc105148947)

[**1.** **Теоретическая часть** 4](#_Toc105148948)

[**1.1** **Компьютерные игры** 4](#_Toc105148949)

[**1.2. Анализ предметной области** 5](#_Toc105148950)

[**1.3 Анализ существующих сред разработки для python** 8](#_Toc105148951)

[**1.4 Выбор стратегии** 12](#_Toc105148952)

[**1.5 Вред азартных игр** 15](#_Toc105148953)

[**2.** **Практическая часть** 16](#_Toc105148954)

[**2.1. Разработка структуры** 16](#_Toc105148955)

[**2.2. Требования к настольному приложению** 16](#_Toc105148956)

[**2.3 Создание приложения и написание кода** 17](#_Toc105148957)

[**Выводы** 38](#_Toc105148958)

[**Список литературы** 40](#_Toc105148959)

[**Приложение 1. Полный код программы** 42](#_Toc105148960)

[**Приложение 2. Руководство пользователя** 57](#_Toc105148961)

# **Введение**

Потребность человека в развлечениях является одной из основных и фундаментальных, а благодаря развитию технологий и компьютеризации мы можем её эффективнее удовлетворять, нам становятся доступны новые формы и способы развлечения.

Цель работы выпускной квалификационной работы является разработка симуляции карточной игры “блэкджек”. Для достижения поставленной цели необходимо решение следующих задач:

1. Разработать логику игры.
2. Создать графический интерфейс.
3. Разработать поведение ИИ.

# **Теоретическая часть**

## **Компьютерные игры**

Для доказательства актуальности выбранной темы, раскрывающую значимость и востребованность компьютерных игр, были рассмотрены статьи Г. П. Кузьминой, И. А. Сидорова - «Компьютерные игры и их влияние на внутренний мир человека» и статья Денисовой А. И. – «Компьютерные игры как феномен современной культуры». В статье «Компьютерные игры и их влияние на внутренний мир человека». Авторы говорят о том, что видеоигры стали важной частью культуры и их можно даже отнести к одному из видов искусства, они (игры) способны не только развлекать, но и приносить определенную пользу так например, игроки в шутеры развивают реакцию. Более того благодаря развитию технологий как внутри игровой индустрии, так и в общем нам становятся доступны новые формы погружения в миры видеоигр. В статье Денисовой компьютерные игры рассматриваются не с позиции влияния на конкретного человека или человечество в целом, а с точки зрения метанарратива культуры иными словами как нечто абстрактное, чем практическое. Она говорит о том, что компьютерные игры являют собой огромный культурный пласт и несут в себе совершенно новые методы восприятия историй и т. д, они позволяют нам взаимодействовать с внутриигровым миром и осознавать себя непосредственным участником происходящих там событий. Более того как уже упоминалось выше игры представляют теперь культурную ценность хотя относительно недавно они котировались исключительно как развлечение для детей. Помимо всего вышеописанного игры развивают технологии и помогают развитию компьютеризации. Эта помощь выражается в виде развития графических технологий и их последующего применения в других областях.

После рассмотрения статей, можно убедиться в не только в актуальности вопроса компьютерных игр, но и в их важности для человеческого общества как с точки зрения развития технологий, так и развития культуры.

## **1.2. Анализ предметной области**

Данная программа представляет собой карточную игру “блэкджек”. Основной целью игры является набор суммы очков превышающий сумму очков дилера, но не больше 21, в противном случае перебор и проигрыш. Каждой карте соответствует определенное число очков – 11 для туза,10 для короля, дамы и валета, у остальных карт значение очков соответствует значению на рубашке.

Далее предлагаю ознакомится с алгоритмом самой игры:

1. Дилер перемешивает все колоды карт.
2. Игроки делают ставки.
3. Дилер всем игрокам раздает по 2 карты, а себе оставляет только одну открытую.
4. Игроки оценивают свои карты и открытую карту дилера.
5. После оценки ситуации игроки могут взять любое количество карт или остаться на той сумме очков что есть. Главное сумма очков не должна превышать “21” очко.
6. Набор карт игроками проходит строго по очереди.
7. После того как все игроки сделали ход, дилер обязан по правилам брать карты если у него 16 и меньше очков и остановиться, когда у него будет 17 и больше очков.

Перед разработкой приложения, нужно определиться с языком программирования, для этого сравним 3 языка и выберем лучший.

1. C++ – Является одни из самых популярных языков, он используется для разработки программного обеспечения. C++ позволяет создавать операционные системы, разнообразные прикладные программы, драйвера для устройств, высокопроизводительных серверов, а также игр.

Преимущества:

* Возможно программировать на низком уровне с памятью, адресами;
* C++ хорошо совместим с языками Си, это позволяет использовать все возможности Си-кода;
* Масштабируемость. На языке C++ разрабатывают программы для самых различных платформ и систем.

Недостатки:

* Строгий синтаксис. Код читается хуже, чем в других языках;
* Сложность в изучении. у С++ сложный синтаксис и маленькая стандартная библиотека, поэтому для изучения с нуля он очень сложен;
* Некоторые преобразования типов неинтуитивны.

1. Python [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) при с [динамической](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [строгой](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%B3%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) типизацией и автоматическим управлением памятью

Преимущества:

* Очень низкий порог вхождения
* Простота написания кода
* Высокая читаемость кода
* Множество модулей и библиотек

Недостатки:

* Низкое быстродействие

1. На JavaScript не получится создать автономное приложение. JavaScript встраивается в HTML-документ, для интерактивности на веб-сайте. С поддержкой JavaScript возможно динамически менять текст, реагировать на события, связанные с действиями пользователя и другое.

Преимущества:

* Без поддержки JavaScript не обойдётся ни один браузер;
* Понятный синтаксис, с написанием плагина или скрипта справится даже новичок;
* Полезные функциональные настройки;
* JavaScript постоянно совершенствуется – Идёт разработка нового проекта, JavaScript2;
* Взаимодействие с приложением может осуществляется даже через текстовые редакторы – Microsoft Office и Open Office.

Недостатки:

* Из-за открытого доступа к исходным кодам скриптов, достаточно низкий уровень безопасности;
* Большое количество мелких ошибок во время работы. Почти все из них легко исправить, но наличие этих ошибок говорит, о том, что этот язык менее профессиональный, сравнительно с другими;
* Некоторая часть приложений, которая активно использует JavaScript, перестанут существовать при отсутствии языка, поскольку полностью основываются на нём.

## **1.3 Анализ существующих сред разработки для python**

До того как, преступить непосредственно к разработке надо определиться с выбором IDE так-как это напрямую влияет на удобство написания кода и общую скорость и эффективность разработки. Основными средами разработки для python являются: PyCharm, Eclipse и Visual Studio рассмотрим каждую из них и сделаем выбор.

IDE (Integrated Development Environment) – интегрированная среда разработки программного обеспечения, комплекс программных средств, используемый программистами для разработки программного обеспечения.

1) Visual Studio — полнофункциональная IDE от Microsoft. Доступная на Windows и Mac OS, Visual Studio поставляется в виде двух версий бесплатной с ограниченным функционалом и платной без всяческих ограничений. Visual Studio позволяет разрабатывать приложения для разных платформ и предоставляет свой собственный набор расширений.

Python Tools for Visual Studio (PTVS) позволяет писать на Python в Visual Studio и включает в себя Intellisense для Python, отладку и другие инструменты.

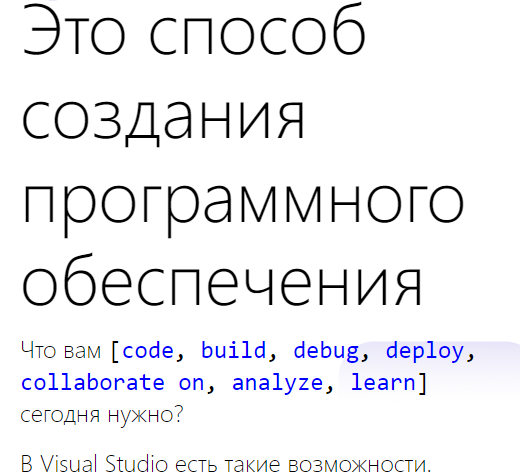


Рис 1. Visual Studio

**Преимущества**:

* Полновесная IDE
* Удобство использования
* Активная поддержка со стороны Microsoft

**Недостатки**:

* Громоздка для созданий маленьких приложений
* Трудна для понимания новичком
* Часто встречаются баги
* Отсутствие поддержки операционных систем Linux

2)PyCharm – Одна из лучших IDE созданных исключительно для Python.

Также как и Visual Studio поставляется в бесплатном и платном виде.

Большим достоинством PyCharm является то что, он поддерживает разработку на Python напрямую — откройте новый файл и начинайте писать код. Вы можете запускать и отлаживать код прямо из PyCharm. Кроме того, в IDE есть поддержка проектов и системы управления версиями.

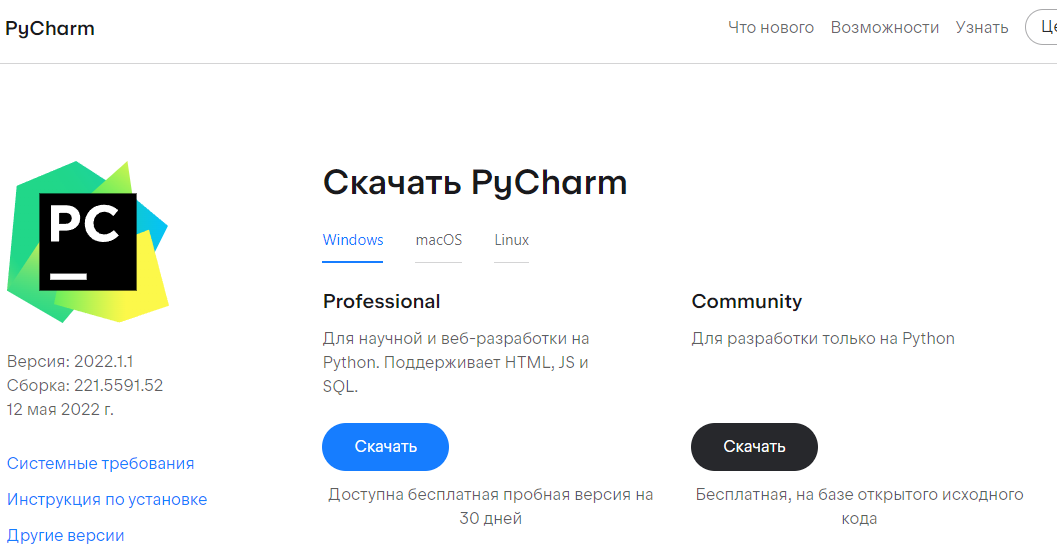


Рис 2. PyCharm

**Преимущества**:

* Ide созданная специально для Python
* Огромное количество встроенных полезных функций
* Удобство и относительная простота использования

**Недостатки**:

* Ввиду тяжеловесности может медленно загружаться на слабых системах

3) Eclipse будучи доступным для Linux, Windows и OS X, Eclipse де-факто является open-source IDE для разработки на Java. Существует множество расширений и аддонов, которые делают Eclipse полезным для разного рода задач.

Одним из таких расширений является PyDev, предоставляющий интерактивную консоль Python и возможности для отладки и автодополнения кода.

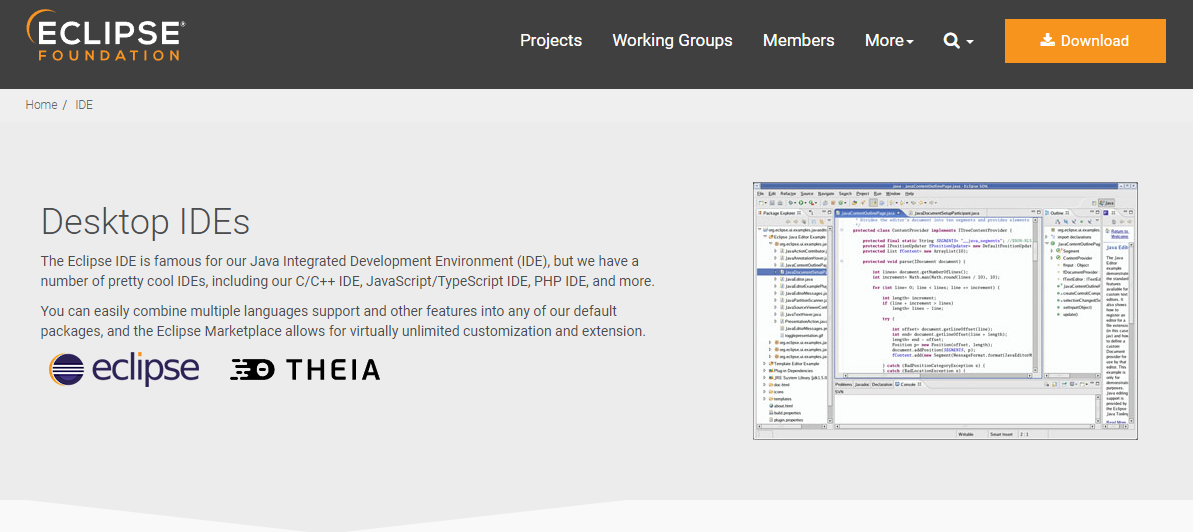


Рис 3. Eclipse

**Преимущества**:

Если у вас уже был установлен Eclipse, то установка PyDev пройдёт быстро и гладко. У опытного пользователя Eclipse не возникнет проблем с изучением этого расширения.

**Недостатки**:

Если вы только начинаете изучать Python или разработку в целом, Eclipse может стать непосильной ношей. IDE большие и требуют больше опыта для полноценного использования в частности это можно сказать про Eclipse.

Из всех просмотренных IDE, PyCharm я считаю лучше. Во-первых: эта IDE специально предназначена для разработки на Python, а во-вторых: она имеет множество полезных при написании кода функций таких как: подсветку синтаксиса, ошибок, авто-дополнение кода, статический анализ кода, широкие возможности отладки и профилирования кода, указание на логические ошибки, присутствуют все необходимые средства для разработки современных приложений.

## **1.4 Выбор стратегии**

Для того чтобы разработать симуляцию карточной игры необходимо прежде всего определиться с тем как будут вести себя игроки, не находящиеся под контролем пользователя иными словами нам необходимо выбрать для них оптимальную для данной программы стратегию. Рассмотрим несколько возможных и выберем подходящую под нашу задачу.

1. Первой возможной стратегией может выступать алгоритм основанный на случайности. Каждый раз, когда перед игроком (неуправляемым пользователем) встает вопрос брать карту или пасовать, происходит бросок “монетки” и в зависимости от него принимается решение.

Достоинства:

Элементарная реализация.

Недостатки:

Крайне неэффективен так-как решение основано на случайности и не учитывает карты на руках и карты дилера.

1. Базовая стратегия – позволяет при каждой конкретной раздаче повысить шансы на выигрыш или минимизировать шансы проиграть.

В отличии от предыдущей (случайной) стратегии, базовая ответ на вопрос брать карту или не брать даёт путем анализа карт на руках

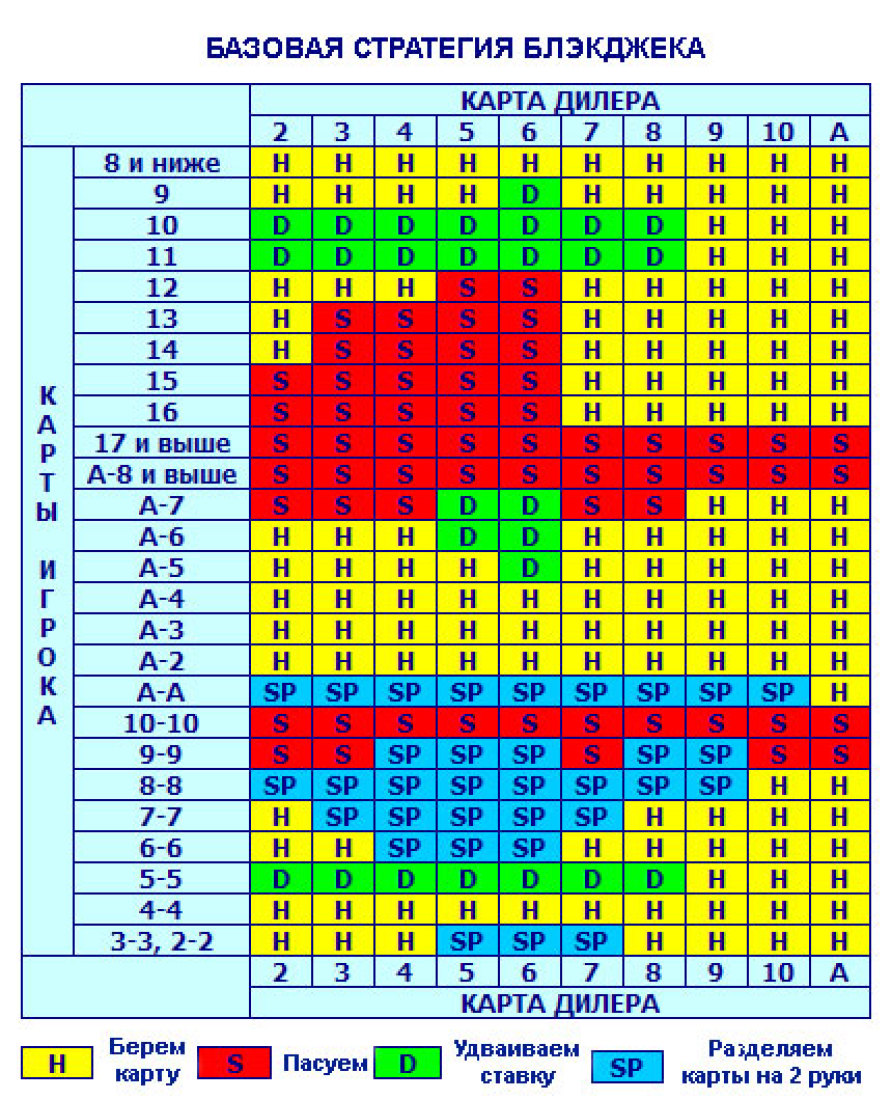
игрока и дилера. (см. рисунок 4 пример базовой стратегии)

Рис 4. Пример базовой стратегии.

Достоинства:

* Простота реализации.
* Позволяет игрокам принимать грамотные решения в зависимости от раздачи.

Недостатки:

* Отсутсвуют

На основе проведенного анализа возможных реализаций алгоритмов поведения игроков самым оптимальным я считаю базовую стратегию так как она отлично справляется с задачей принятия решений и как следствие делает игровой процесс интереснее.

## **1.5 Вред азартных игр**

Учитывая специфику разрабатываемой игры считаю важным уделить внимание вопросу опасности азартных игр к которым и относиться “блэкджек”. Для начала рассмотрим само понятие азартной игры.

Азартные игры можно определить, как ставку нечто ценного на определенный случайный результат т.е независящий полностью от навыков самого игрока. Из определения напрямую следует то что играя в подобные игры можно уйти в минус, попасть в долги и остаться без средств на существование. Кроме того, увлечение подобными развлечениями имеет неочевидные последствия для здоровья такие как: Мигрени, высокое давление, болезни сердца и даже кожи. Из всего вышесказанного стоит сделать вывод о том, что к азартным играм надо подходит с большой долей осторожности и внимательности, играть только на виртуальные деньги (которые не обладают ценностью в реальной жизни) и не увлекаться.

# **Практическая часть**

## **2.1. Разработка структуры**

Поскольку интерес людей к цифровым развлечениям в общем и к играм в частности растёт с каждым годом это рождает необходимость его удовлетворять.

Для решения этой проблемы необходимо создавать приложения с использованием современных технологий способных удовлетворить этот спрос.

Главными эффектами от разработки подобных приложений выступают:

* Удовлетворение потребности в развлечениях;
* Увеличение рабочих мест так-как для создания игр нужны специалисты;

Главной целью дипломной работы является создание карточной игры «блэкджек».

Можно выделить следующие цели решения задачи:

* Удовлетворение потребности пользователей в досуге;
* Повышение интереса к цифровым развлечениям;

Симуляция карточной игры, должна обеспечивать выполнение следующих основных функций:

* Обеспечить опыт идентичный опыту при реальной игре;
* Создать поведенческие алгоритмы противников игрока, дилера;
* Предоставить удобный пользовательский интерфейс;

## **2.2. Требования к настольному приложению**

Программа должна соответствовать следующим требованиям:

1. Иметь простой запуск с помощью файла с расширением EXE;
2. Алгоритмы поведения противников и дилера на основе стратегии;
3. Простой, понятный и удобный интерфейс.

## **2.3 Создание приложения и написание кода**

Для начала создадим проект в PyCharm.

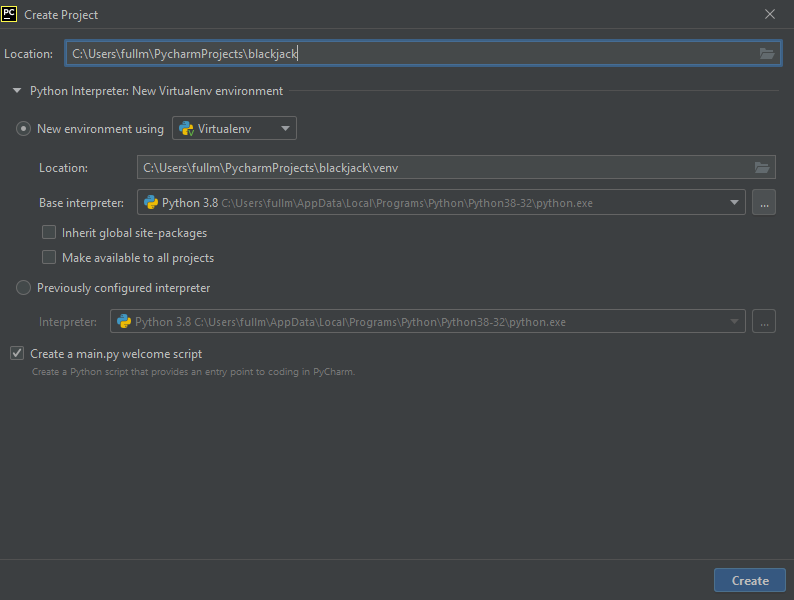


Рис 4. Создание проекта

Теперь после того как проект создан можем приступить к написанию кода, для начала импортируем нужные для работы библиотеки: tkinter - для создания GUI (Graphical User Interface), PIL - для работы с JPG, random и метод shuffle- для перемешивания карт.

Листинг 1.

import tkinter as tk

from tkinter.ttk import Combobox

import tkinter.messagebox

from PIL import Image, ImageTk

import random

После того как все необходимые библиотеки импортированы можно приступать к написанию вспомогательных классов, для начала создадим класс Card в котором будет хранится информация о карте.

Листинг 2.

class Card():

#каждый экземпляр класса будет иметь значение и масть(value,suit)

def \_\_init\_\_(self, value, suit):

self.value = value

self.suit = suit

#каждой карте в игре соответствует числовое значение данный метод позволяет его получить

def get\_card\_value(self):

if self.value in 'JQK':

return 10

elif self.value == 'A':

return 11

else:

return '23456789'.index(self.value)+2

#функция для вывода значения и масти

def \_\_str\_\_(self):

return (self.value + self.suit)

Теперь создадим класс Hand который будет представлять собой набор карт каждого игрока и дилера.

Листинг 3.

class Hand():

#класс Hand состоит из экземпляров класса Card которые он хранит в списке cards

def \_\_init\_\_(self):

self.cards = []

#функция добавляет в список новую карту

def add\_card(self,card):

self.cards.append(card)

#функция позволяет получить суммарное числовое значение всех карт и их количество

def get\_value(self):

res = 0

for card in self.cards:

if(card.get\_card\_value() == 'A' and res <= 21):

res += card.get\_card\_value()

elif(card.get\_card\_value() == 'A' and res > 21):

res += 1

else:

res += card.get\_card\_value()

return res

#функция для вывод информации о картах

def \_\_str\_\_(self):

res = ""

for card in self.cards:

res += str(card) + " "

res += str(self.get\_value())

return res

После этого напишем класс Deck который будет создавать колоду карт и перемешивать её.

Листинг 4.

class Deck():

def \_\_init\_\_(self):

values = '23456789JQKA'

suits = 'SCDH'

self.cards = [Card(v,s) for v in values for s in suits]

random.shuffle(self.cards)

#функция позволяет получить карту и удаить её из колоды или проще говоря сдать карту

def deal\_card(self):

return self.cards.pop()

После создания вспомогательных классов перейдем к основному классу игры Game в нём будет прописана основная логика игры, создан графический интерфейс и задано поведение ИИ. Первым делом при создании объекта класса будет задана рабочая область и создано окно меню.

Листинг 5.

root = tk.Tk()#создаём рабочую область

def \_\_init\_\_(self):

self.root = Game.root

self.root.geometry('640x480')#задаём размеры окна

#открываем изображение и с помощью метода resize библиотеки Image растягиваем изображение меню по размеру окна

img = Image.open('images/menubackground.jpg')

imag = img.resize((640, 480), Image.ANTIALIAS)

image = ImageTk.PhotoImage(imag)#этот метод позволяет использовать изображение tkinter

panel = tk.Label(self.root, image=image)

panel.pack(side="top", fill="both", expand="no")#pack размещает изобржаение в качестве фона

#создаём две кнопки первая при нажатии очищает все виджеты и переносит в меню выбора кол-во игроков, вторая закрывает окно

self.b1 = tk.Button(self.root, text='начать игру', command=lambda: [self.b1.place\_forget(),self.b2.place\_forget(),Game.players\_num(self)],bg="black", fg="orange")

self.b2 = tk.Button(self.root, text='выйти', command=self.root.quit, bg="black", fg="orange")

#функции для размещения кнопок

self.b1.place(x=250, y=220)

self.b2.place(x=250, y=260)

#запускает окно и выполняет вышеописанные действия

self.root.mainloop()

Cледом идёт функция player’s\_num которая вызывается в случае если пользователь нажал в меню кнопку начать игру и предлагает выбрать количество игроков

Листинг 6.

def players\_num(self):

#нажатие в предыдущей функции кнопки "начать игру" запускает эту функцию

#с помощью combobox пользователь выбирает количество игроков

combo = Combobox()

combo['values'] = (1, 2, 3)

combo.current(1)

combo.place(x=250, y=240)

#после выбора количества игроков пользователь нажимает кнопку "начать игру и запускается основаная функция игры board"

lbl = tk.Label(text="выберете кол-во игроков",font = ("Arial Bold",10),bg="black", fg="orange")

b3 = tk.Button(text='начать игру', command=lambda:[b3.place\_forget(), combo.place\_forget(), lbl.place\_forget(),Game.board(combo.get())], bg="black", fg="orange")

lbl.place(x=250, y=220)

b3.place(x=250, y=260)

После выбора количества игроков и подтверждения (нажатием на кнопку “начать игру”) будет вызвана board которая создаёт игровое поле, а также в ней с помощью вспомогательных функций отрисовываются карты и задаётся поведение ИИ. Первым делом после вызова удаляются старые виджеты и создаётся окно.

Листинг 7.

def board(self,pn):

#основная функция игры

#этот цикл удалет все предыдущие виджеты

for widget in self.root.winfo\_children():

widget.destroy()

#открываем изображение игрового поля и ставим его на фон

img = Image.open('images/boardbg.jpg')

imag = img.resize((640, 480), Image.ANTIALIAS)

image = ImageTk.PhotoImage(imag)

panel = tk.Label(self.root,image=image)

panel.pack(side="top", fill="both", expand="no")

Затем создаётся колода и происходит раздача и отрисовка карт с помощью вспомогательных классов.

Листинг 8.

#создание колоды

d = Deck()

#список для сохранения ссылок на использованные изображения карт

#(если его не использоваться будет отрисовываться только последняя)

self.pics = []

#создание списка для хранения экземпляров класса Hand() для каждого игрока

players\_hand = []

#создание экземпляров класса Hand()

for i in range(int(pn)):

players\_hand.append(Hand())

#создание экземпляра класса Hand() для дилера

dealer\_hand = Hand()

for i in range(1,len(players\_hand)):

#в зависимости от номера игрока даёт ему две карты, располагает их на нужных позициях и отрисовывает

if (i == 1): x, y = 85, 205

if (i == 2): x, y = 500, 205

players\_hand[i].add\_card(d.deal\_card())

players\_hand[i].add\_card(d.deal\_card())

#список карт для текущего игрока(нужен для правильной передачи информации для отрисовки)

cardlist = players\_hand[i].\_\_str\_\_().split(sep=' ')

for j in range(len(cardlist) - 1):

self.count = 0

Game.drawpic(cardlist[j], x, y)

x += 20

#раздаёт пользователю две карты

players\_hand[0].add\_card(d.deal\_card())

players\_hand[0].add\_card(d.deal\_card())

#раздаёт дилеру две карты

dealer\_hand.add\_card(d.deal\_card())

#список карт пользователя(нужен для правильной передачи информации для отрисовки)

cardlist = players\_hand[0].\_\_str\_\_().split(sep=' ')

self.x,self.y = 296,270#задаёт координаты для отрисовки карт пользователя

#отрисовка карт пользователя

for i in range(len(cardlist)-1):

Game.drawpic(cardlist[i],self.x,self.y)

self.x += 20

#задаёт координаты для отрисовки карт дилера

self.x,self.y = 296,50

#отрисовка карт дилера

Game.drawpic(dealer\_hand.\_\_str\_\_()[:2],self.x,self.y)

#понадобится позднее для проверки условий

in\_game = True

self.x,self.y = 316,270#задаёт координаты для отрисовки карт пользователя

Рассмотрим более подробно функцию drawpic которая отвечает за прорисовку карт

Листинг 9.

def drawpic(self,card,x,y):

#функция поолучая информацию о карте(значаение и масть) с помощью функции cropp отрисовывает изображение карты на игровом поле

self.image = ImageTk.PhotoImage(Game.cropp(card).resize((60, 90), Image.ANTIALIAS))

panel1 = tk.Label(self.root, image=self.image)

self.pics.append(self.image)

panel1.place(x=x, y=y)

Как вы могли заметить она использует функцию cropp которая вырезает из картинки с картами в файлах игры нужную. Посмотрим как она это делает

Листинг 10.

def cropp(self,card):

#функция которая получая данные карты(значение и масть) открывает файл со всеми картами и ищет нужную

img = Image.open('images/cards.jpg')

if('S' in card):

x1, x2, x3, x4 = 15, 40, 820, 144

elif ('H' in card):

x1, x2, x3, x4 = 15, 125, 820, 228

elif('C' in card):

x1, x2, x3, x4 = 15, 213, 820, 315

elif ('D' in card):

x1, x2, x3, x4 = 15, 300, 820, 400

if '2' in card:

cropped = img.crop((x1+67, x2, x3-685, x4-22))

elif '3' in card:

cropped = img.crop((x1+129, x2, x3-622, x4-22))

elif '4' in card:

cropped = img.crop((x1+193, x2, x3-560, x4-22))

elif '5' in card:

cropped = img.crop((x1+254, x2, x3-499, x4-22))

elif '6' in card:

cropped = img.crop((x1+316, x2, x3-436, x4-22))

elif '7' in card:

cropped = img.crop((x1+378, x2, x3-373, x4-22))

elif '8' in card:

cropped = img.crop((x1+439, x2, x3-312, x4-22))

elif '9' in card:

cropped = img.crop((x1+500, x2, x3-251, x4-22))

elif '10' in card:

cropped = img.crop((x1+563, x2, x3-188, x4-22))

elif 'J' in card:

cropped = img.crop((x1+624, x2, x3-126, x4-22))

elif 'Q' in card:

cropped = img.crop((x1+687, x2, x3-64, x4-22))

elif 'K' in card:

cropped = img.crop((x1+747, x2, x3-3, x4-22))

elif 'A' in card:

cropped = img.crop((x1+6, x2, x3-745, x4-22))

self.cropped = cropped

return self.cropped

Теперь вернёмся к функции board после раздачи и отрисовки карт игроки начинают ходить

Листинг 11.

while (players\_hand[0].get\_value() < 21):

#пока сумма очков карт < 21 пользователю предлагается выбор взять карту или нет

q = tk.messagebox.askquestion('', 'взять карту(д/н)?')

if (q == 'yes'):

#если пользователь выбирает взять,то алгоритм выдает её ему и отрисовывает на игровом поле

self.x += 20

players\_hand[0].add\_card(d.deal\_card())

cardlist = players\_hand[0].\_\_str\_\_().split(sep=' ')

Game.drawpic(cardlist[len(cardlist)-2],self.x,self.y)

if (players\_hand[0].get\_value() > 21):

#выполняется в случае если сумма очков больше 21

tk.messagebox.showinfo('', 'вы проиграли!')

if(int(pn) > 1): break #если игроков > 1 то ход переходит к ним

else: in\_game = False #иначе если игорк 1(т.е только пользователь и дилер то игра заканчивается)

else:

#в случае если пользователь отказался брать карту то ход переходит к другим

tk.messagebox.showinfo('', 'вы не взяли карту')

break

После хода игрока ходят противники, решение взять или не взять карту принимается с помощью вызова функции choice которая основываясь на карте дилера и текущей сумме очков на руках игрока принимает решение.

Листинг 12.

def choice(self,psum,dsum):

#функция которая в зависимости от значений карт на руках и дилера определяет брать боту карту или нет

if(psum == 12 and (dsum in (5,6))):

return 'no'

elif(psum in (13,14) and dsum in (3,4,5,6)):

return 'no'

elif(psum in (15,16) and dsum in (2,3,4,6)):

return 'no'

elif(psum >= 17):

return 'no'

else:

return 'yes'

Алгоритм хода противников

Листинг 13.

for i in range(1,len(players\_hand)):#цикл для хода ботов(игроков которыми пользователь не управляет)

#в зависимости от того какой именно бот задаются координаты для отрисовки

if (i == 1):

x, y = 125, 205

if (i == 2):

x, y = 540, 205

while (players\_hand[i].get\_value() < 21):

q = Game.choice(players\_hand[i].get\_value(),dealer\_hand.get\_value())#вызов функции choice

#для выбор брать или не брать карту

if (q == 'yes'):

tk.messagebox.showinfo('', 'игорк ' + str((i+1)) + ' взял карту')

players\_hand[i].add\_card(d.deal\_card())

cardlist = players\_hand[i].\_\_str\_\_().split(sep=' ')

Game.drawpic(cardlist[len(cardlist)-2],x,y)

x+= 20

if (players\_hand[i].get\_value() > 21):

tk.messagebox.showinfo('', 'игорк ' + str((i+1)) + ' проиграл!')

break

else:

tk.messagebox.showinfo('', 'игорк '+str((i+1)) +' не взял карту')

break

После хода ботов игра проверяет остались ли не проигравшие игроки(сумма очков которых меньше 21) и в случае положительного ответа дилер начинает набирать карты пока сумма очков на его руках меньше 17

Листинг 14.

if (in\_game):

#если игра идёт и не выбыли все игорки то дилер набирает карты пока сумма их очков < 17

self.x, self.y = 316, 50

while (dealer\_hand.get\_value() < 17):

dealer\_hand.add\_card(d.deal\_card())

dealerlist = list(dealer\_hand.\_\_str\_\_().split(sep=' '))

Game.drawpic(dealerlist[len(dealerlist)-2],self.x,self.y)#отрисовка взятой карты

self.x += 20

if (dealer\_hand.get\_value() > 21):

#если сумма очков дилера > 21 игра заканчивается его поражением

tk.messagebox.showinfo('', 'дилер проиграл!')

in\_game = False

После чего программа проверяет если у дилера не было перебора (т.е сумма очков меньше 21) и остался хотя-бы один игрок и в случае выполнения этих условий идёт подсчёт сумм очков на руках у игроков и дилера победителем считается игрок набравший больше всех очков

Листинг 15.

if (in\_game):

#если остался дилер и не все игроки выбыли то создаём словарь элемент которого состоит из номера игрока и суммы его очков

slist = {}

for i in range(len(players\_hand)):

if players\_hand[i].get\_value() <= 21: slist['игорк ' + str(i+1)] = players\_hand[i].get\_value()#если сумма очков > 21то

#значение в словарь не добавляется

slist['dealer'] = dealer\_hand.get\_value()

# находит максимальное значение в словаре

maxs = max(slist.values())

winner = 0

for k, v in slist.items():

if v == maxs:

winner = k

tk.messagebox.showinfo('', winner + ' победил набрав больше всего очков (' + str(maxs) + ')')

#и выводит номер победителя с сообщением о победе

После определения победителя игры заканичивается и пользователю предоставляется выбор выйти или сыграть ещё одну игру, если пользователь выбирает последний вариант, то все виджеты удаляются и вызывается класс Game()

Листинг 16.

q1 = tk.messagebox.askquestion('', 'сыграть ещё(д/н)?')#выбор сыграть ещё или нет

if(q1 == 'yes'):

#если выбрано сыграть то удаляются все виджеты с окана и вызывается класс Game

for widget in self.root.winfo\_children():

widget.destroy()

Game()

else:

#в случае нет то окно игры закрывается

exit(0)

# **Выводы**

В результате проделанной работы была создана симуляция карточной игры “блэкджек”, а также сделаны следующие выводы:

1. Сделан обзор темы компьютерных игр.
2. Разработана игра с основанным на базовой стратегии блэкджека поведением игроков.
3. Создан графический интерфейс.
4. Разработано руководство пользователя.

# **Список литературы**

1. Денисова А. И. – «Компьютерные игры как феномен современной культуры»: https://cyberleninka.ru/article/n/kompyuternye-igry-kak-fenomen-sovremennoy-kultury/viewer
2. Г. П. Кузьмина, И. А. Сидоров - «Компьютерные игры и их влияние на внутренний мир человека»: https://cyberleninka.ru/article/n/kompyuternye-igry-i-ih-vliyanie-na-vnutrenniy-mir-cheloveka
3. Рик Гаско. Простой Python с нуля. Солон-Пресс: 2019
4. С.К. Буйначев, Н.Ю. Боклаг. Python. Основы программирования. Учебное пособие Издательство Уральского Университета.: Екатеринбург, 2014.
5. Документация для модуля “tkinter” [Электронный ресурс].

https://tkdocs.com/index.html

1. Дэвид Бизли. Python. Подробный справочник 4-ое издание.

Символ-плюс, 2010.

1. Сайт с описанием базовой стратегии [Электронный ресурс].

https://game-wiki.guru/published/blekdzhek/bazovaya-strategiya-blekdzheka.html

1. Сайт правил игры в “блэкджек” [Электронный ресурс].

https://add-hobby.ru/blackjack.html

1. Сайт Visual Studio [Электронный ресурс]. https://visualstudio.microsoft.com/ru/downloads/
2. Сайт PyCharm [Электронный ресурс]. https://www.jetbrains.com/ru-ru/pycharm/download/
3. Сайт Eclipse [Электронный ресурс]. https://www.eclipse.org

# **Приложение 1. Полный код программы**

Листинг 17.

#импортируем библиотеки

#tkinter - для создания gui,PIL - для работы с JPG,random - для перемешивания карт

import tkinter as tk

from tkinter.ttk import Combobox

import tkinter.messagebox

from PIL import Image, ImageTk

import random

class Card():

#каждый экземпляр класса будет иметь значение и масть(value,suit)

def \_\_init\_\_(self, value, suit):

self.value = value

self.suit = suit

#каждой карте в игре соответствует числовое значение данный метод позволяет его получить

def get\_card\_value(self):

if self.value in 'JQK':

return 10

elif self.value == 'A':

return 11

else:

return '23456789'.index(self.value)+2

#функция для вывода значения и масти

def \_\_str\_\_(self):

return (self.value + self.suit)

class Hand():

#класс Hand состоит из экземпляров класса Card которые он хранит в списке cards

def \_\_init\_\_(self):

self.cards = []

#функция добавляет в список новую карту

def add\_card(self,card):

self.cards.append(card)

#функция позволяет получить суммарное числовое значение всех карт и их количество

def get\_value(self):

res = 0

for card in self.cards:

if(card.get\_card\_value() == 'A' and res <= 21):

res += card.get\_card\_value()

elif(card.get\_card\_value() == 'A' and res > 21):

res += 1

else:

res += card.get\_card\_value()

return res

#функция для вывод информации о картах

def \_\_str\_\_(self):

res = ""

for card in self.cards:

res += str(card) + " "

res += str(self.get\_value())

return res

#класс Deck создаёет колоду карт и перемешивает её с помощью метода shuffle

class Deck():

def \_\_init\_\_(self):

values = '23456789JQKA'

suits = 'SCDH'

self.cards = [Card(v,s) for v in values for s in suits]

random.shuffle(self.cards)

#функция позволяет получить карту и удаить её из колоды или проще говоря сдать карту

def deal\_card(self):

return self.cards.pop()

#основной класс программы

class Game:

root = tk.Tk()#создаём рабочую область

def \_\_init\_\_(self):

self.root = Game.root

self.root.geometry('640x480')#задаём размеры окна

#открываем изображение и с помощью метода resize библиотеки Image растягиваем изображение меню по размеру окна

img = Image.open('images/menubackground.jpg')

imag = img.resize((640, 480), Image.ANTIALIAS)

image = ImageTk.PhotoImage(imag)#этот метод позволяет использовать изображение tkinter

panel = tk.Label(self.root, image=image)

panel.pack(side="top", fill="both", expand="no")#pack размещает изобржаение в качестве фона

#создаёем две кнопки первая при нажатии очищает все виджеты и переносит в меню выбора кол-во игроков,вторая закрывает окно

self.b1 = tk.Button(self.root, text='начать игру', command=lambda: [self.b1.place\_forget(),self.b2.place\_forget(),Game.players\_num(self)],bg="black", fg="orange")

self.b2 = tk.Button(self.root, text='выйти', command=self.root.quit, bg="black", fg="orange")

#функции для размещения кнопок

self.b1.place(x=250, y=220)

self.b2.place(x=250, y=260)

#запускает окно и выполняет вышеописаные действия

self.root.mainloop()

@staticmethod

def players\_num(self):

#нажатие в предыдущей функции кнопки "начать игру" запускает эту функцию

#с помощью combobox пользователь выбирает количество игроков

combo = Combobox()

combo['values'] = (1, 2, 3)

combo.current(1)

combo.place(x=250, y=240)

#после выбора количества игроков пользователь нажимает кнопку "начать игру и запускается основаная функция игры board"

lbl = tk.Label(text="выберете кол-во игроков",font = ("Arial Bold",10),bg="black", fg="orange")

b3 = tk.Button(text='начать игру', command=lambda:[b3.place\_forget(), combo.place\_forget(), lbl.place\_forget(),Game.board(combo.get())], bg="black", fg="orange")

lbl.place(x=250, y=220)

b3.place(x=250, y=260)

@classmethod

def cropp(self,card):

#функция которая получая данные карты(значение и масть) открывает файл со всеми картами и ищет нужную

img = Image.open('images/cards.jpg')

if('S' in card):

x1, x2, x3, x4 = 15, 40, 820, 144

elif ('H' in card):

x1, x2, x3, x4 = 15, 125, 820, 228

elif('C' in card):

x1, x2, x3, x4 = 15, 213, 820, 315

elif ('D' in card):

x1, x2, x3, x4 = 15, 300, 820, 400

if '2' in card:

cropped = img.crop((x1+67, x2, x3-685, x4-22))

elif '3' in card:

cropped = img.crop((x1+129, x2, x3-622, x4-22))

elif '4' in card:

cropped = img.crop((x1+193, x2, x3-560, x4-22))

elif '5' in card:

cropped = img.crop((x1+254, x2, x3-499, x4-22))

elif '6' in card:

cropped = img.crop((x1+316, x2, x3-436, x4-22))

elif '7' in card:

cropped = img.crop((x1+378, x2, x3-373, x4-22))

elif '8' in card:

cropped = img.crop((x1+439, x2, x3-312, x4-22))

elif '9' in card:

cropped = img.crop((x1+500, x2, x3-251, x4-22))

elif '10' in card:

cropped = img.crop((x1+563, x2, x3-188, x4-22))

elif 'J' in card:

cropped = img.crop((x1+624, x2, x3-126, x4-22))

elif 'Q' in card:

cropped = img.crop((x1+687, x2, x3-64, x4-22))

elif 'K' in card:

cropped = img.crop((x1+747, x2, x3-3, x4-22))

elif 'A' in card:

cropped = img.crop((x1+6, x2, x3-745, x4-22))

self.cropped = cropped

return self.cropped

@classmethod

def drawpic(self,card,x,y):

#функция поолучая информацию о карте(значаение и масть) с помощью функции cropp отрисовывает изображение карты на игровом поле

self.image = ImageTk.PhotoImage(Game.cropp(card).resize((60, 90), Image.ANTIALIAS))

panel1 = tk.Label(self.root, image=self.image)

self.pics.append(self.image)

panel1.place(x=x, y=y)

@classmethod

def choice(self,psum,dsum):

#функция которая в зависимости от значений карт на руках и дилера определяет брать боту карту или нет

if(psum == 12 and (dsum in (5,6))):

return 'no'

elif(psum in (13,14) and dsum in (3,4,5,6)):

return 'no'

elif(psum in (15,16) and dsum in (2,3,4,6)):

return 'no'

elif(psum >= 17):

return 'no'

else:

return 'yes'

@classmethod

def board(self,pn):

#основная функция игры

#этот цикл удалет все предыдущие виджеты

for widget in self.root.winfo\_children():

widget.destroy()

#открываем изобржаение игрового поля и ставим его на фон

img = Image.open('images/boardbg.jpg')

imag = img.resize((640, 480), Image.ANTIALIAS)

image = ImageTk.PhotoImage(imag)

panel = tk.Label(self.root,image=image)

panel.pack(side="top", fill="both", expand="no")

#создание колоды

d = Deck()

#список для сохранения ссылок на использованные изображения карт

#(если его не использоваться будет ортрисовываться только последняя)

self.pics = []

#создание списка для хранения экземпляров класса Hand() длякаждого игрока

players\_hand = []

#создание экземпляров класса Hand()

for i in range(int(pn)):

players\_hand.append(Hand())

#создание экземпляра класса Hand() для дилера

dealer\_hand = Hand()

for i in range(1,len(players\_hand)):

#в зависимости от номера игрока даёт ему две карты,располагает их на нужных позициях и отрисовывает

if (i == 1): x, y = 85, 205

if (i == 2): x, y = 500, 205

players\_hand[i].add\_card(d.deal\_card())

players\_hand[i].add\_card(d.deal\_card())

#список карт для текущего игрока(нужен для правильной передачи информации для отрисовки)

cardlist = players\_hand[i].\_\_str\_\_().split(sep=' ')

for j in range(len(cardlist) - 1):

self.count = 0

Game.drawpic(cardlist[j], x, y)

x += 20

#раздаёт пользователю две карты

players\_hand[0].add\_card(d.deal\_card())

players\_hand[0].add\_card(d.deal\_card())

#раздаёт дилеру две карты

dealer\_hand.add\_card(d.deal\_card())

#список карт пользователя(нужен для правильной передачи информации для отрисовки)

cardlist = players\_hand[0].\_\_str\_\_().split(sep=' ')

self.x,self.y = 296,270#задаёт координаты для отрисовки карт пользователя

#отрисовка карт пользователя

for i in range(len(cardlist)-1):

Game.drawpic(cardlist[i],self.x,self.y)

self.x += 20

#задаёт координаты для отрисовки карт дилера

self.x,self.y = 296,50

#отрисовка карт дилера

Game.drawpic(dealer\_hand.\_\_str\_\_()[:2],self.x,self.y)

#понадобится позднее для проверки условий

in\_game = True

self.x,self.y = 316,270#задаёт координаты для отрисовки карт пользователя

#цикл игры

while (players\_hand[0].get\_value() < 21):

#пока сумма очков карт < 21 пользователю предлагается выбор взять карту или нет

q = tk.messagebox.askquestion('', 'взять карту(д/н)?')

if (q == 'yes'):

#если пользователь выбирает взять,то алгоритм выдает её ему и отрисовывает на игровом поле

self.x += 20

players\_hand[0].add\_card(d.deal\_card())

cardlist = players\_hand[0].\_\_str\_\_().split(sep=' ')

Game.drawpic(cardlist[len(cardlist)-2],self.x,self.y)

if (players\_hand[0].get\_value() > 21):

#выполняется в случае если сумма очков больше 21

tk.messagebox.showinfo('', 'вы проиграли!')

if(int(pn) > 1): break #если игроков > 1 то ход переходит к ним

else: in\_game = False #иначе если игорк 1(т.е только пользователь и дилер то игра заканчивается)

else:

#в случае если пользователь отказался брать карту то ход переходит к другим

tk.messagebox.showinfo('', 'вы не взяли карту')

break

for i in range(1,len(players\_hand)):#цикл для хода ботов(игроков которыми пользователь не управляет)

#в зависимости от того какой именно бот задаются координаты для отрисовки

if (i == 1):

x, y = 125, 205

if (i == 2):

x, y = 540, 205

while (players\_hand[i].get\_value() < 21):

q = Game.choice(players\_hand[i].get\_value(),dealer\_hand.get\_value())#вызов функции choice

#для выбор брать или не брать карту

if (q == 'yes'):

tk.messagebox.showinfo('', 'игорк ' + str((i+1)) + ' взял карту')

players\_hand[i].add\_card(d.deal\_card())

cardlist = players\_hand[i].\_\_str\_\_().split(sep=' ')

Game.drawpic(cardlist[len(cardlist)-2],x,y)

x+= 20

if (players\_hand[i].get\_value() > 21):

tk.messagebox.showinfo('', 'игорк ' + str((i+1)) + ' проиграл!')

break

else:

tk.messagebox.showinfo('', 'игорк '+str((i+1)) +' не взял карту')

break

if (in\_game):

#если игра идёт и не выбыли все игорки то дилер набирает карты пока сумма их очков < 17

self.x, self.y = 316, 50

while (dealer\_hand.get\_value() < 17):

dealer\_hand.add\_card(d.deal\_card())

dealerlist = list(dealer\_hand.\_\_str\_\_().split(sep=' '))

Game.drawpic(dealerlist[len(dealerlist)-2],self.x,self.y)#отрисовка взятой карты

self.x += 20

if (dealer\_hand.get\_value() > 21):

#если сумма очков дилера > 21 игра заканчивается его поражением

tk.messagebox.showinfo('', 'дилер проиграл!')

in\_game = False

if (in\_game):

#если остался дилер и не все игроки выбыли то создаём словарь элемент которого состоит из номера игрока и суммы его очков

slist = {}

for i in range(len(players\_hand)):

if players\_hand[i].get\_value() <= 21: slist['игорк ' + str(i+1)] = players\_hand[i].get\_value()#если сумма очков > 21то

#значение в словарь не добавляется

slist['dealer'] = dealer\_hand.get\_value()

# находит максимальное значение в словаре

maxs = max(slist.values())

winner = 0

for k, v in slist.items():

if v == maxs:

winner = k

tk.messagebox.showinfo('', winner + ' победил набрав больше всего очков (' + str(maxs) + ')')

#и выводит номер победителя с сообщением о победе

q1 = tk.messagebox.askquestion('', 'сыграть ещё(д/н)?')#выбор сыграть ещё или нет

if(q1 == 'yes'):

#если выбрано сыграть то удаляются все виджеты с окана и вызывается класс Game

for widget in self.root.winfo\_children():

widget.destroy()

Game()

else:

#в случае нет то окно игры закрывается

exit(0)

self.root.mainloop()#для работы и отображения виджетов tkinter

g1 = Game()#первый вызов класса Game

# **Приложение 2. Руководство пользователя**

После открытия программы перед пользователем появляется меню.

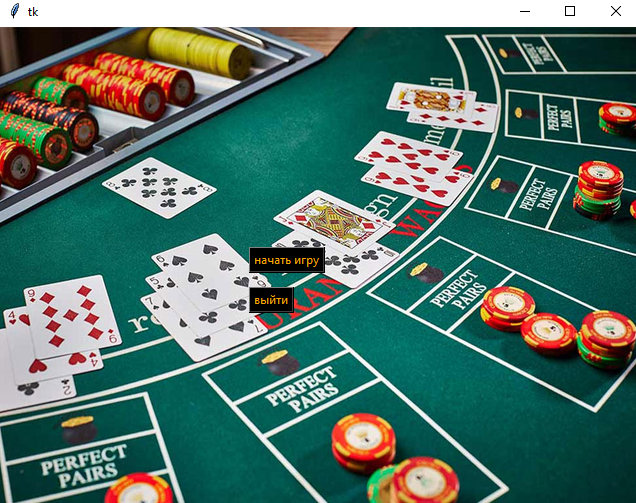


Рис 5. Главное меню

В случае выбора ответа выйти программа просто закроет окно. В случае выбора варианта начать игру мы увидим меню выбора количества игроков (по умолчанию стоит значение 2)

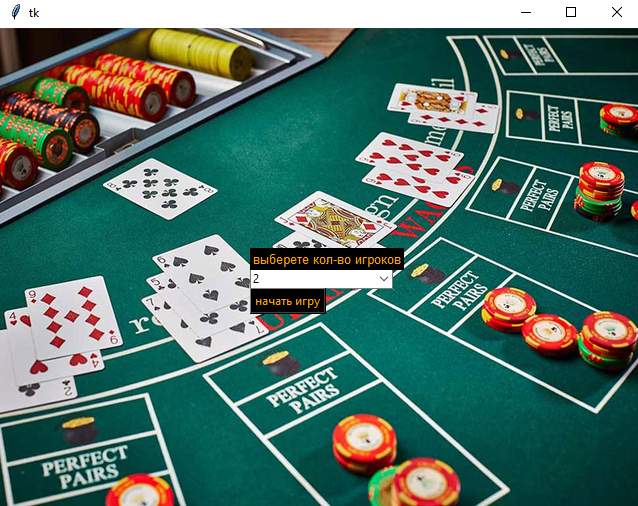


Рис 6. Выбор количество игроков

После выбора нужного значения и нажатия на кнопку “начать игру” появится игровое поле с отрисованными картами



Рис 7. Игровое поле

Пользователю будет предложен выбор взять карту или нет, в случае положительного ответа будет отрисованна ещё одна карта и также она будет добавлена в руку игрока.

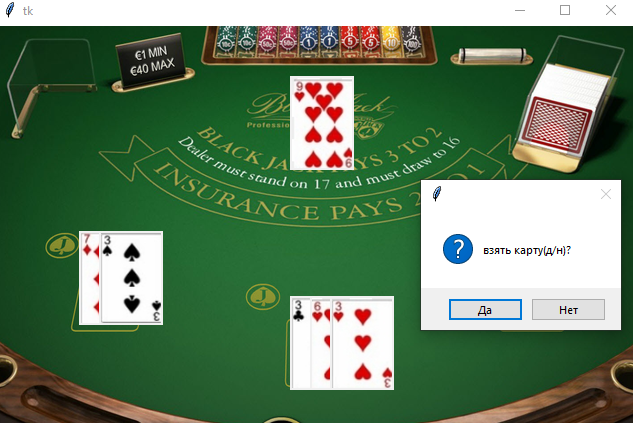


Рис 8. Пополнение руки игрока

Предложение взять карту будет появляться до момента отказа или пока сумма не превысит 21 очко.



Рис 9. Перебор карт игроком

В случае отказа от взятия карты ход перейдёт к другому игроку который на основании имеющихся у него и у дилера карт примет решение брать или не брать карту в случае выбора варианта о взятии карты она отрисовывется и добавляется в руку.

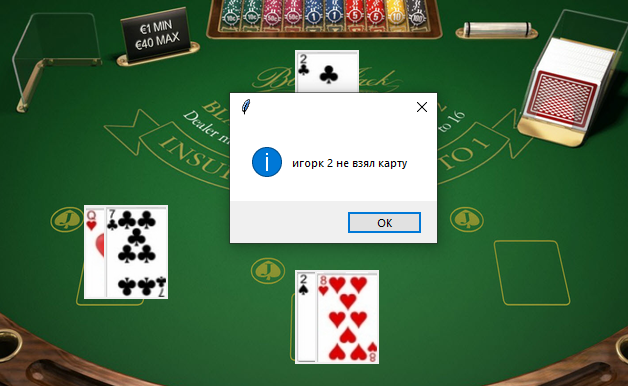


Рис 10. Ход противника

После того как каждый из противников отказался брать карту или проиграл карты начинает набирать дилер (происходит мгновенно) и определяется победитель.

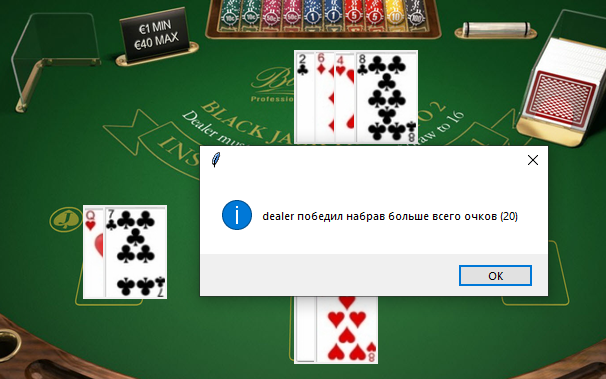


Рис 11. Конец игры

После определения победителя игроку даётся выбор сыграть ещё одну партию или нет, в случае выбора второго варианта окно закрывается, а выбор первого возвращает на окно главного меню



Рис 12. Играть ещё партию или нет