



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
(ДИПЛОМНЫЙ ПРОЕКТ)**

На тему: Создание игры на объектно-ориентированном языке Python

по специальности: 09.02.07 Информационные системы и программирование

Выполнил

Д.О. Фоменко

Руководитель

Ю.В. Жирнова

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора по информационным технологиям

_____ / А.В.Капанова /

« ____ » _____ 2022г.

Содержание

Введение	3
Теоретическая часть	4
Компьютерные игры	4
1.2. Анализ предметной области	5
1.3 Анализ существующих сред разработки для python	7
1.4 Выбор стратегии	11
1.5 Вред азартных игр	13
Практическая часть	14
2.1. Разработка структуры	14
2.2. Требования к настольному приложению	14
2.3 Создание приложения и написание кода	15
Выводы	41
Список литературы	41
Приложение 1. Полный код программы	
Приложение 2. Руководство пользователя	

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.

Обучение с подкреплением штурмом взяло мир искусственного интеллекта (ИИ). Начиная от AlphaGo и AlphaStar, все большее число видов деятельности, в которых раньше доминировал человек, теперь завоевано агентами ИИ, работающими на основе обучения с подкреплением. Эти достижения зависят от оптимизации действий агента в определенной среде для достижения максимального вознаграждения. Чтобы продемонстрировать, как искусственный интеллект работает на практике в задаче оценки различных значений состояния, было решено провести пошаговую демонстрацию на примере игры в блекджек.

Цель работы выпускной квалификационной работы является создание игры на объектно-ориентированном языке Python. Для достижения поставленной цели необходимо решение следующих задач:

1. Сделать обзор темы компьютерных игр.
2. Разработать логику игры.
3. Создать графический интерфейс.
4. Разработать поведение ИИ.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 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 котиrowались 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.

В статье “Всё, что вам нужно знать об ИИ — за несколько минут”, описывается, что ИИ является мощным средством обработки данных и может находить решения сложных задач быстрее, чем традиционные алгоритмы, написанные программистами. ИНС и методики глубокого обучения могут помочь решить ряд разнообразных проблем. Минус состоит в том, что самые оптимизированные модели часто работают как «чёрные ящики», не давая возможности изучить причины выбора ими того или иного решения. Этот факт может привести к этическим проблемам, связанным с прозрачностью информации.

После рассмотрения статей, можно убедиться в не только в актуальности вопроса компьютерных игр, но и в том, что для их обучения необходимо использовать ИИ.

1.2. Анализ предметной области

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

1) C++ – Является один 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 C++ сложный синтаксис и маленькая стандартная библиотека, поэтому для изучения с нуля он очень сложен;

- Некоторые преобразования типов неинтуитивны.

2) Python высокоуровневый язык программирования при с динамической строгой типизацией и автоматическим управлением памятью

Преимущества:

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

Недостатки:

- Низкое быстродействие

3) 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.

Подводя итоги, для совмещения визуализации игры и ИИ оптимально выбрать язык программирования Python, так как он предоставляет необходимые библиотеки.

1.3 Анализ существующих сред разработки для python

До того как приступить 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.

Это способ создания программного обеспечения

Что вам [`code`, `build`, `debug`, `deploy`,
`collaborate on`, `analyze`, `learn`]
сегодня нужно?

В Visual Studio есть такие возможности.

Рисунок 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.


PyCharm

Что нового

Возможности

Узнать

Ц



Версия: 2022.1.1
Сборка: 221.5591.52
12 мая 2022 г.

[Системные требования](#)
[Инструкция по установке](#)
[Другие версии](#)

Скачать PyCharm

[Windows](#)[macOS](#)[Linux](#)

Professional

Для научной и веб-разработки на Python. Поддерживает HTML, JS и SQL.

[Скачать](#)

Доступна бесплатная пробная версия на 30 дней

Community

Для разработки только на Python

[Скачать](#)

Бесплатная, на базе открытого исходного кода

Рисунок 2. PyCharm

Преимущества:

- Иде 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.

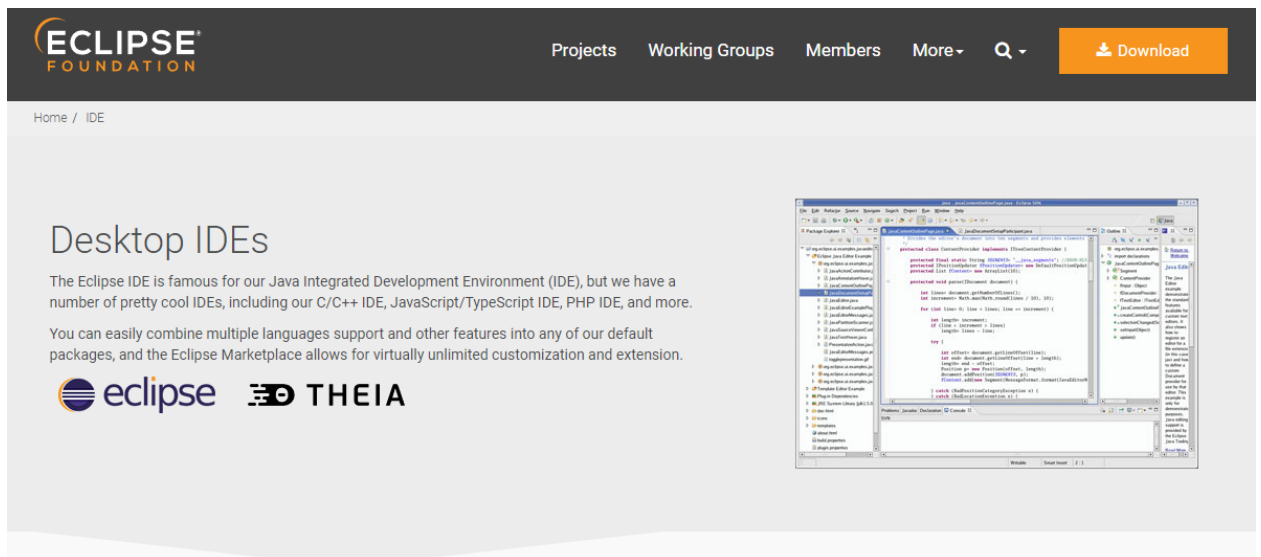


Рисунок 3. Eclipse

Преимущества:

Если у вас уже был установлен Eclipse, то установка PyDev пройдет 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 наилучшим образом подходит для выполнения поставленной цели. 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, автодополнение 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.

2) 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)

Достоинства:

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

Недостатки:

- Отсутствуют

БАЗОВАЯ СТРАТЕГИЯ БЛЭКДЖЕКА

		КАРТА ДИЛЕРА									
		2	3	4	5	6	7	8	9	10	A
КАРТЫ ИГРОКА	8 и ниже	H	H	H	H	H	H	H	H	H	H
	9	H	H	H	H	D	H	H	H	H	H
	10	D	D	D	D	D	D	D	H	H	H
	11	D	D	D	D	D	D	D	H	H	H
	12	H	H	H	S	S	H	H	H	H	H
	13	H	S	S	S	S	H	H	H	H	H
	14	H	S	S	S	S	H	H	H	H	H
	15	S	S	S	S	S	H	H	H	H	H
	16	S	S	S	S	S	H	H	H	H	H
	17 и выше	S	S	S	S	S	S	S	S	S	S
	A-8 и выше	S	S	S	S	S	S	S	S	S	S
	A-7	S	S	S	D	D	S	S	H	H	H
	A-6	H	H	H	D	D	H	H	H	H	H
	A-5	H	H	H	H	D	H	H	H	H	H
	A-4	H	H	H	H	H	H	H	H	H	H
	A-3	H	H	H	H	H	H	H	H	H	H
	A-2	H	H	H	H	H	H	H	H	H	H
	A-A	SP	SP	SP	SP	SP	SP	SP	SP	SP	H
	10-10	S	S	S	S	S	S	S	S	S	S
	9-9	S	S	SP	SP	SP	S	SP	SP	S	S
	8-8	SP	SP	SP	SP	SP	SP	SP	SP	H	H
	7-7	H	SP	SP	SP	SP	SP	H	H	H	H
	6-6	H	H	SP	SP	SP	H	H	H	H	H
	5-5	D	D	D	D	D	D	D	H	H	H
	4-4	H	H	H	H	H	H	H	H	H	H
	3-3, 2-2	H	H	H	SP	SP	SP	H	H	H	H
		2	3	4	5	6	7	8	9	10	A
		КАРТА ДИЛЕРА									

H Берем карту
 S Пасуем
 D Удваиваем ставку
 SP Разделяем карты на 2 руки

Рисунок 4. Пример базовой стратегии.

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 “блэджек”. 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.

Кроме того, разработка данной игры покажет, что вероятность выигрыша игрока программируется и не является случайным числом. С помощью симуляции большого количества игр было выявлено, какое преимущество есть у казино. Вот что удалось обнаружить:

Казино получает конкурентное преимущество, заставляя игроков совершать действия до дилера (действовать в условиях неполной информации). Это создает риск получить перебор (таким образом игроки теряют все еще до того, как дилер совершает хоть какое-нибудь действие).

Особую опасность представляют ситуации, когда карты на руках игроков стоят от 12 до 16 очков (в таком случае вероятность получить перебор со следующей картой максимальная), а дилер показывает старшую карту. В таких случаях можно предположить, что у дилера будет более высокое значение, так что игрокам остается только брать еще или проигрывать. Вероятность выигрыша или ничьей к сумме карт игрока

Наконец, простейшая «наивная» стратегия брать карты в случае нулевой вероятности получить перебор значительно увеличивает шансы на победу, ведь в таком случае растет вероятность того, что проиграет казино.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Разработка структуры

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 ознакомиться с алгоритмом самой игры:

Дилер перемешивает все колоды карт.

Игроки делают ставки.

Дилер всем игрокам раздает по 2 карты, а себе оставляет только одну открытую.

Игроки оценивают свои карты и открытую карту дилера.

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.

Набор карт игроками проходит строго по очереди.

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.

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.

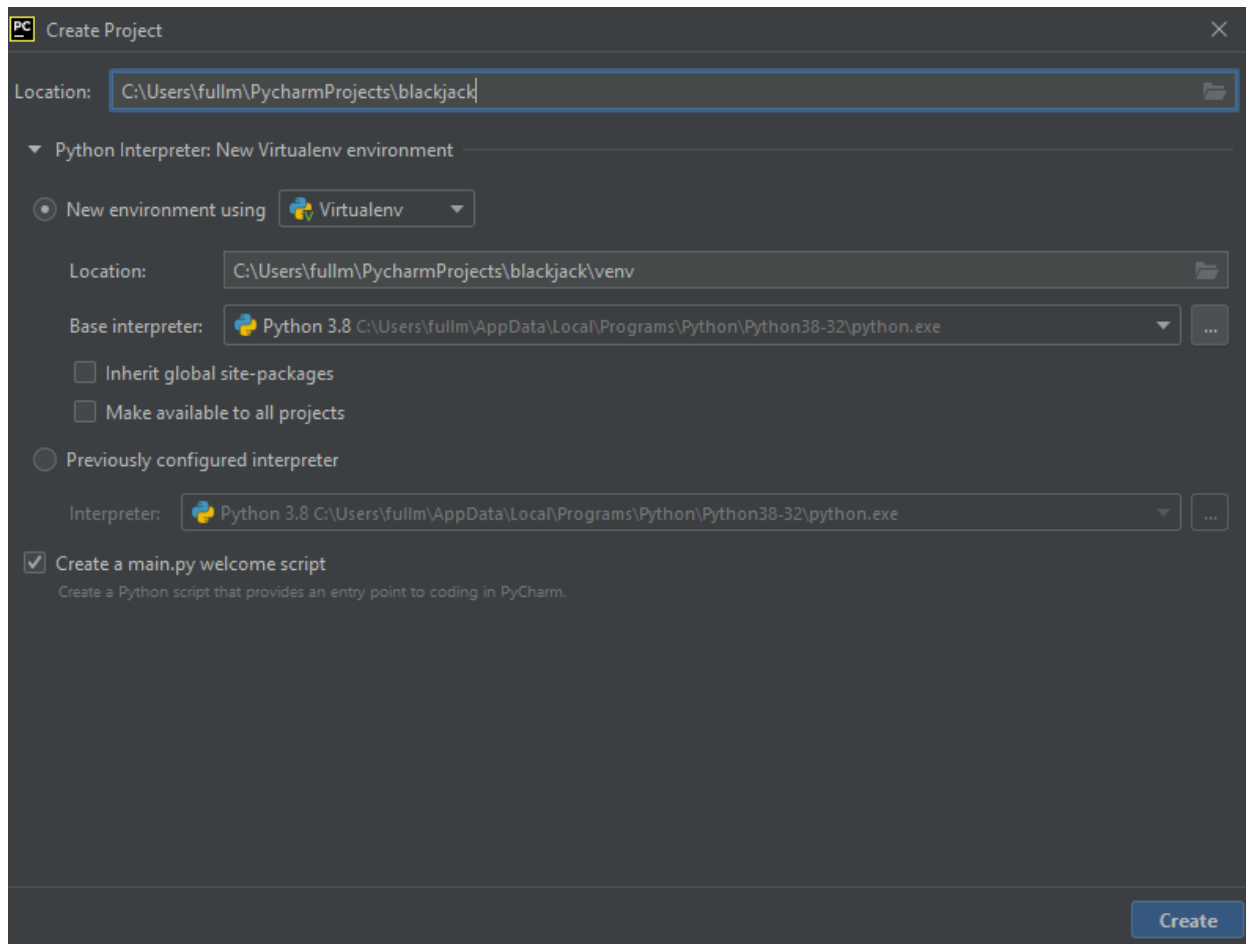


Рисунок 4. Создание проекта

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 приступить к написанию вспомогательных классов, для начала создадим класс Card в котором будет храниться 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 удалить 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
    изображение 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()

```

Следом 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 основная 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 получая информацию о карте(значение и масть) с помощью функции сторр 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 игрок 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(", 'игрок ' + 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(", 'игрок ' + str((i+1)) + ' ' +  
проиграл!')
```

```
    break
```

```
else:
```

```
    tk.messagebox.showinfo(", 'игрок ' + 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 игроки 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['игрок ' + 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 заканчивается 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)
```

Полный код программы представлен на сайте
<https://github.com/bastix321/Diplom.git>

2.4. Руководство пользователя

После открытия программы перед пользователем появляется меню.



Рисунок 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)



Рисунок 6. Выбор количество игроков

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

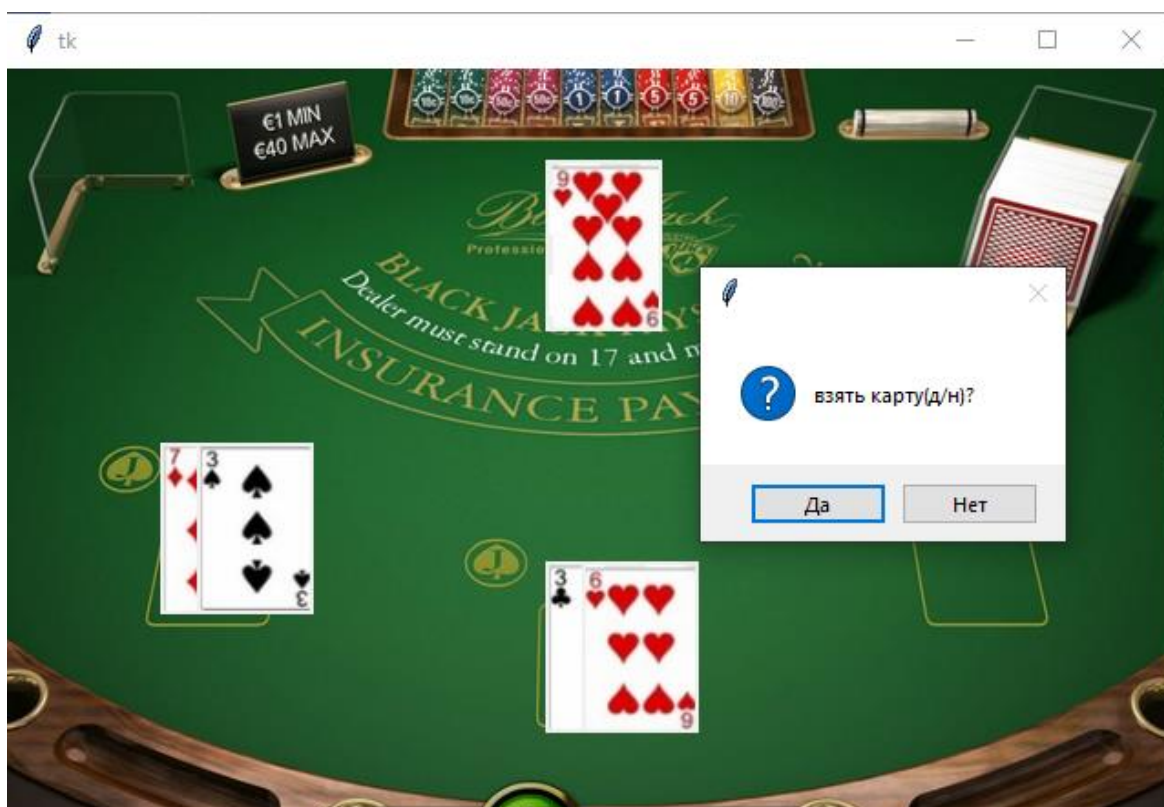


Рисунок 7. Игровое поле

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.



Рисунок 8. Пополнение руки игрока

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



Рисунок 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 отрисовывается и добавляется в руку.



Рисунок 10. Ход противника

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

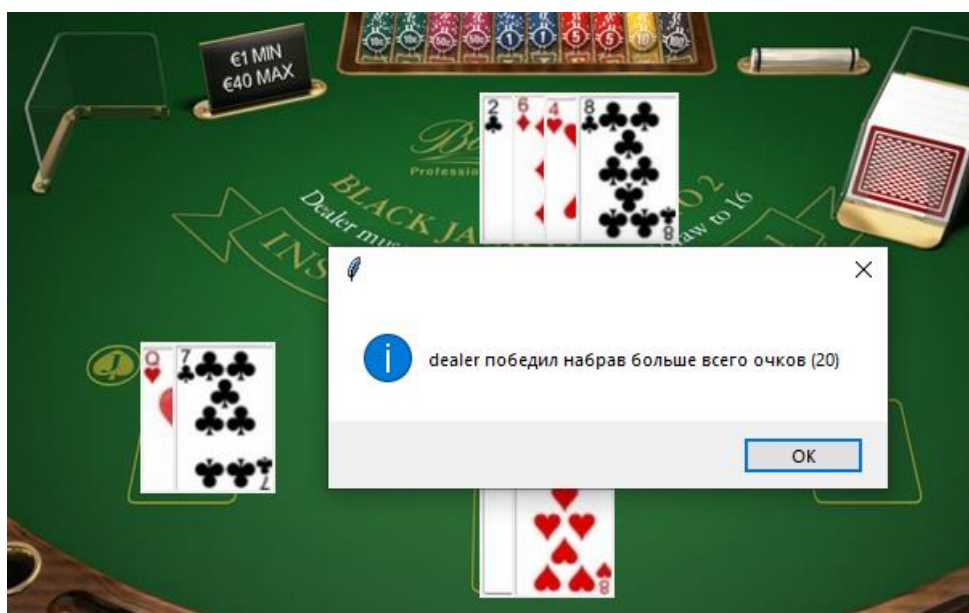


Рисунок 11. Конец игры

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



Рисунок 12. Играть ещё партию или нет

2.5. Публикация игры

Теперь выложим игру на сайт <https://itch.io/> для этого пройдем регистрацию.

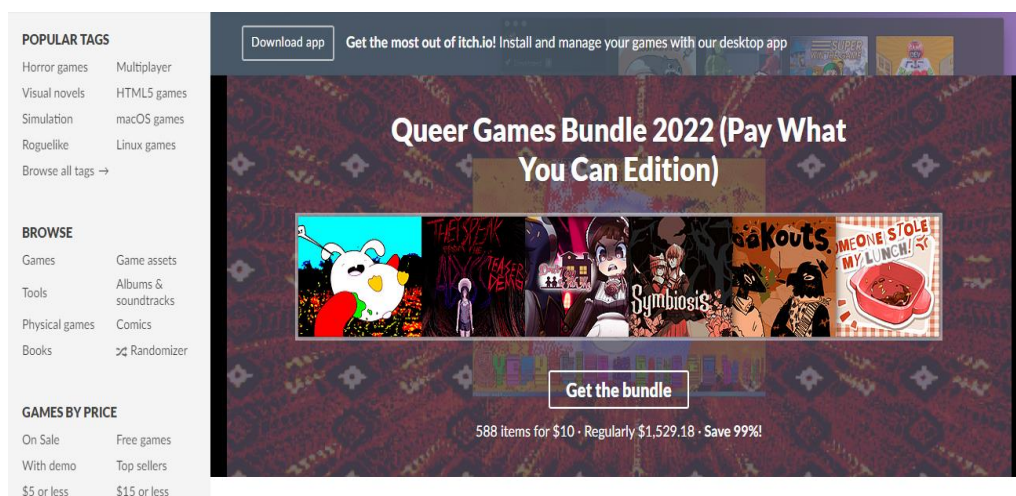


Рис 13. Главная страница сайта

После регистрации выбираем пункт Upload new project, открывается окно с меню создания проекта.

The screenshot shows the 'Create a new project' interface on Itch.io. At the top is a 'Dashboard' header. Below it is the title 'Create a new project'. A yellow warning banner states: 'You don't have payment configured. If you set a minimum price above 0 no one will be able to download your project. [Edit account](#)'. The form is divided into two columns. The left column contains: a tip 'Make sure everyone can find your page' with a link to 'quality guidelines'; a 'Title' input field; a 'Project URL' input field containing 'https://brusilov.itch.io/'; a 'Short description or tagline' input field with the placeholder 'Optional'. The right column features a large dashed box for a cover image with a red 'Upload Cover Image' button, and a section for 'Gameplay video or trailer' with a note to provide a link to YouTube or Vimeo.

Dashboard

Create a new project

You don't have payment configured. If you set a minimum price above 0 no one will be able to download your project. [Edit account](#)

Make sure everyone can find your page
Review our [quality guidelines](#) before posting your project

Title

Project URL

Short description or tagline
Shown when we link to your project. Avoid duplicating your project's title

Gameplay video or trailer
Provide a link to YouTube or Vimeo.

The cover image is used whenever itch.io wants to link to your project from another part of the site. Required (Minimum: 315x250, Recommended: 630x500)

Рис 14. Меню создания проекта

Вводим название в нашем случае Blackjack и выбираем нужные пункты.

Title

Blackjack

Project URL

https://brusilov.itch.io/blackjack

Short description or tagline
Shown when we link to your project. Avoid duplicating your project's title

Optional

Classification
What are you uploading?

Games — A piece of software you can play ▼

Kind of project

Downloadable — You only have files to be downloaded ▼

TIP You can add additional downloadable files for any of the types above

Release status

Released — Project is complete, but might receive some updates ▼

Рис 15. Выбранные пункты

После того как мы задали имя и указали общую информацию об игре, загрузим файл установщика.

Uploads

main.exe

More... Delete file

9mb · [Change display name](#)

Executable ▾ for ☒ Windows ☐ Linux ☐ macOS ☐ Android

☐ Hide this file and prevent it from being downloaded

TIP Use **butler** to upload game files: it only uploads what's changed, generates patches for the [itch.io app](#), and you can automate it. [Get started!](#)

Upload files

or

[Add External file](#) ?

File size limit: 1 GB. [Contact us](#) if you need more space

Рис. 17 Загрузка установщика

И наконец настроим доступ к проекту и возможность обсуждения.

Community

Build a community for your project by letting people post to your page.

- ☒ Disabled
- ☐ Comments — Add a nested comment thread to the bottom of the project page
- ☐ Discussion board — Add a dedicated community page with categories, threads, replies & more

Visibility & access

Use Draft to review your page before making it public. [Learn more about access modes](#)

- ☐ Draft — Only those who can edit the project can view the page
- ☐ Restricted — Only owners & authorized people can view the page
- ☒ Public — Anyone can view the page

Public access settings ×

☐ Disable new downloads & purchases ?

☐ Unlisted in search & browse

Рис 18. Настройка доступа и обсуждения

После всех вышеописанных действий нажимаем кнопку “сохранить и посмотреть страницу” и попадаем на окно нашего уже выложенного проекта (<https://brusilov.itch.io/blackjack> - ссылка на проект).

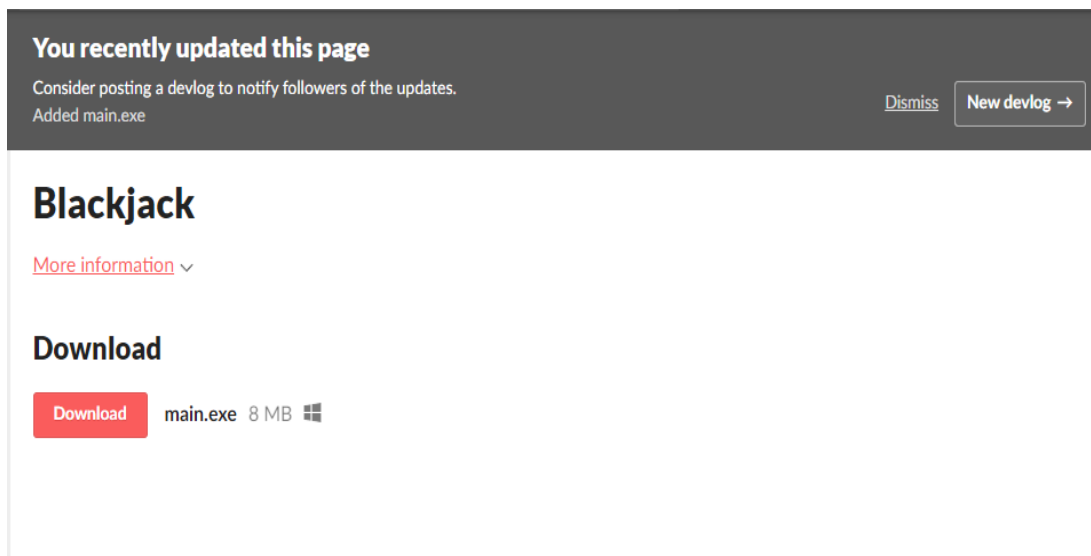


Рис 19. Страница игры

ЗАКЛЮЧЕНИЕ

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

1. Сделан обзор темы компьютерных игр.
2. Разработана логика игры.
3. Создан графический интерфейс.
4. Разработано поведение ИИ.

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>
6. Дэвид Бизли. Python. Подробный справочник 4-ое издание. Символ-плюс, 2010.
7. Сайт с описанием базовой стратегии [Электронный ресурс]. <https://game-wiki.guru/published/blekdzhek/bazovaya-strategiya-blekdzheka.html>
8. Сайт правил игры в “блэкджек” [Электронный ресурс]. <https://add-hobby.ru/blackjack.html>
9. Сайт Visual Studio [Электронный ресурс]. <https://visualstudio.microsoft.com/ru/downloads/>
10. Сайт PyCharm [Электронный ресурс]. <https://www.jetbrains.com/ru-ru/pycharm/download/>
11. Сайт Eclipse [Электронный ресурс]. <https://www.eclipse.org>