МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«МУРМАНСКИЙ АРКТИЧЕСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «МАГУ»)**

Факультет МАТЕМАТИКИ, ЭКОНОМИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра МАТЕМАТИКИ, ФИЗИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

выпускная квалификационная работа

**РАЗРАБОТКА ИНТЕРАКТИВНОГО ПРИЛОЖЕНИЯ ДЛЯ ДИАГНОСТИКИ ПСИХОЛОГТЧЕСКИХ ОСОБЕННОСТЕЙ ДЕТЕЙ**

|  |
| --- |
| Выполнила обучающийся: |
| Сумина Дарья Александровна, |
| 02.03.01 Математика и компьютерные науки, |
| направленность (профиль) Математика и |
| компьютерные науки, очная форма обучения |
| (Ф.И.О., специальность /направление подготовки, направленность (профиль), форма обучения) |
| Научный руководитель: |
| Ляш Олег Иванович, канд. пед. наук, доцент |
| доцент кафедры математики, физики и |
| информационных технологий |
| (ФИО, ученая степень, звание, должность) |

Мурманск

2019

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc11092369)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ПСХОЛОГИЧЕСКОГО ТЕСТИРОВАНИЯ С ПОМОЩЬЮ ПРИЛОЖЕНИЙ 5](#_Toc11092370)

[1.1 Основные определения 5](#_Toc11092371)

[1.2 Методики психологического тестирования 6](#_Toc11092372)

[1.3 Описание и оценка имеющихся аналогов 7](#_Toc11092373)

[1.4 Требования 9](#_Toc11092374)

[1.5 Методы решения 9](#_Toc11092375)

[ГЛАВА 2. РЕАЛИЗАЦИЯ ИНТЕРАКТИВНОГО ПРИЛОЖЕНИЯ ДЛЯ ДИАГНОСТИКИ ПСИХОЛОГИЧЕСКИХ ОСОБЕННОСТЕЙ ДЕТЕЙ 12](#_Toc11092376)

[2.1 Постановка задачи 12](#_Toc11092377)

[2.2 Выбор и описание средств разработки 12](#_Toc11092378)

[2.3 Описание функционала предлагаемого решения 14](#_Toc11092379)

[2.4 Функциональная модель данных 17](#_Toc11092380)

[2.5 Необходимые для реализации библиотеки и модули 19](#_Toc11092381)

[2.6 Проектирование пользовательского интерфейса 20](#_Toc11092382)

[ЗАКЛЮЧЕНИЕ 40](#_Toc11092383)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 41](#_Toc11092384)

[ПРИЛОЖЕНИЯ 45](#_Toc11092385)

# **ВВЕДЕНИЕ**

В настоящее время человек не представляет своей жизни без компьютерных технологий. Диагностика и обучение с помощью программного продукта помогает ускорить процесс и освободить от персонального участия психолога, проанализировать результаты в разы быстрее, уменьшить ошибки по причине невнимательности специалиста (человеческий фактор) и сделать работу психолога более продуктивной.

Однако не всегда можно найти материал или подходящую программу. В сети Интернет довольно много различных программ и тестов психологического содержания. Но к сожалению, большинство из них платные. Рынок программного обеспечения психологических тестов дошкольников недостаточно развит: демоверсии либо отсутствуют, либо недостаточно информативны. В большинстве случаев нельзя сделать вывод об их пригодности для деятельности конкретного детского психолога. Те тесты, что можно проходить в режиме онлайн, не ориентированы на детей дошкольного возраста, которые в силу возраста требуют помощи взрослых для результативного прохождения теста.

**Актуальность** заключается в ускорении процесса диагностики, анализа результатов, уменьшения ошибок и в возможности сделать работу более продуктивной.

Также актуальность работы подтверждается наличием заказа от «Центра психолого-педагогической, медицинской и социальной помощи».

**Целью** работы является разработка интерактивного приложения для диагностики психологических особенностей детей.

**Объектом** исследования является психологическое тестирование.

**Предмет –** автоматизация процесса психологического тестирования с помощью программ.

Для достижения цели работы необходимо решить следующие **задачи**:

1. Изучить литературу по теме работы.
2. Проанализировать существующие аналоги.
3. Сформулировать требования к собственной разработке.
4. Сделать обоснованный выбор средств разработки.
5. Выполнить проектирование программных средств.
6. Выполнить программную реализацию.

**Методы** исследования: общенаучные методы анализа и синтеза, компьютерный эксперимент, методы системного анализа, объектно-ориентированного анализа и моделирование.

**Новизна** заключается в создании приложения для анализа формирования предпосылок эмоционального интеллекта с учетом специфики возраста тестируемых.

**Апробация**

Выступление на региональной научно-практической конференции «Современные информационные технологии и естественные науки» 15-20 апреля 2019 года (Приложение 1).

Выпускная квалификационная работа имеет следующую **структуру:** введение; две главы основной части (первая – теоретические аспекты психологического тестирования с помощью приложений; вторая – реализация интерактивного приложения для диагностики психологических особенностей детей); заключение; список использованных источников; приложение.

Количество страниц работы 91, 40 пунктов списка использованных источников, количество таблиц 16, количество рисунков 15. Основной тест содержит 40 страниц, приложение – 51 страниц.

# **ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ПСХОЛОГИЧЕСКОГО ТЕСТИРОВАНИЯ С ПОМОЩЬЮ ПРИЛОЖЕНИЙ**

* 1. **Основные определения**

Особую роль в развитии дошкольника играют эмоции и чувства. Чувства проявляются в разнообразных эмоциях – кратковременных и ситуативных переживаниях [7].

Эмоции – совокупность психологических явлений, представляющих состояние человека, связанные с процессом удовлетворения потребностей [22].

Для определения того, какие из известных свойств насколько выражены у конкретного человека, используются современные психологические тесты [29].

Тестами в психологии называют стандартизированные методики психодиагностики, позволяющие получать сопоставимые количественные и качественные показатели степени развитости изучаемых свойств [23].

Существует много разновидностей тестов, которые разделяются на группы по нескольким основаниям: по предмету тестирования; по особенностям используемых в тесте задач; по материалу, предъявляемому испытуемым; по объекту оценивания [23].

Тестирование – процесс применения тестов и других психодиагностических методик для изучения и оценки психологии и поведения человека [23].

Термин «диагностика» происходит от греческих слов dia – между, через, после и gnosis – знание, познание, понятие. Diagnostikos – распознавать. В литературе можно встретить различные толкования данного термина. Однако во всех случаях под словом «диагностика» понимают такую деятельность, которая осуществляется на основе полученного знания о человеке, коллективе, явлении, процессе и заключает в себе соответствующие рекомендации [24].

* 1. **Методики психологического тестирования**

Психологическое тестирование достаточно хорошо разработано методически. Рассмотрим несколько примеров.

*Цветовой тест Люшера* представляет собой цветовой тест, созданный швейцарским психологом. Тест состоит из четырёх основных и четырех дополнительных цветов. В основные цвета входят синий, сине-зеленый, оранжево-красный и светло-желтый. Синий цвет символизирует спокойствие и удовлетворенность; сине-зеленый – упрямство, настойчивость, чувство уверенности; оранжево-красный – возбуждение, агрессивность, силу волевого усилия; светло-желтый – стремление к общению, веселость, активность.

В дополнительные цвета входят фиолетовый, коричневый, черный, нулевой. Эти цвета символизируют негатив: огорчение, страх, тревожность, раздражительность. При тестировании получаются восемь позиций. Первые две означают сильное предпочтение, вторые две – предпочтение, третьи вторые – безразличие к цвету, последние же – антипатию. На основе полученных результатов программа делает примерную характеристику испытуемого.

*Тест Роршаха* – методика личности, разработанная немецким ученым Роршахом для изучения неосознаваемых склонностей и побуждений человека [23]. Тест состоит из 10 карточек, половина из который черно-белая, другая – цветная. На каждой из карточек чернилами нанесен симметричный рисунок, который мало напоминает что-то конкретное. Испытуемый должен рассказать и максимально детализировать увиденное на изображении. А психолог на основе услышанного делает умозаключение. Этот тест также можно проходить с помощью компьютера. Главное строить анализ на четырех аспектах: ориентирование в пространстве, избирательность восприятия, динамичность или неподвижность образов, очерёдность реакции.

*Тест Спилбергера-Ханина* представляет собой тест из сорока вопросов, где нужно выбрать один из четырёх ответов (почти никогда, никогда, часто, почти всегда). Данный тест отражает ситуационную и личную тревожность в структуре личности тестируемого на данное время. При анализе следует учесть, что итоговый показатель по каждой из шкал (шкала личной тревожности и шкала ситуативной тревожности) находится в диапозоне от 20 до 80 баллов. Причем чем выше результат, тем выше уровень тревожности.

Приведем сравнительную таблицу представленных тестов (Таблица 1).

Таблица 1 – Сравнительная таблица

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Разновидность теста | Характерные особенности | Доступ |
| Цветовой тест Люшера | Селективный (изобритательный) тест | Тестируемый выбирает один вариант из нескольких | Открытый |
| Тест Роршаха | Ассоциативно-образный тест | Тестируемый обозначает ассоциации, возникающие у него при ответе на определенную карточку | Открытый |
| Тест Спилбергера-Ханна | Закрытый тест | Ответы тестируемого ограничены несколькими вариантами (почти никогда, никогда, часто, почти всегда) | Открытый |

На основе рассмотренных широко известных примеров можно убедится, что методически психологическое тестирование разработано хорошо.

* 1. **Описание и оценка имеющихся аналогов**

Программ для тестирования довольно много. Однако наибольший интерес для нас представляют программы для психодиагностики детей дошкольного возраста или программы-тренажеры для распознавания эмоций. Рассмотрим несколько примеров.

*«1С:Дошкольная психодиагностика. Базовая версия»* представляет собой однопользовательскую программу, предназначенную для психологов дошкольных образовательных учреждений. Программа включает в себя различный набор методик, основанный на тесте с выбором одного ответа. Продукт способен учитывать, сохранять и анализировать результаты в зависимости от того, кто проходил тест – ребёнок, группа детей или родители. В состав программы входят методики, предназначенный для оценок познавательной сферы, социально-психологической адаптации в коллективе, креативности, эмоционального состояния, нейро-психологического состояния, произвольности, индивидуальных особенностей ребенка.

*Программа «Мимика» 3.20.* Это тренажер для распознавания разных эмоций. Лица людей выражают семь эмоций (радость, грусть, гнев, удивление, отвращение, страх и презрение). Эмоции выражены на фотографиях людей в профиль, фас и с поворотом головы. Программа помогает научиться распознавать микровыражения по фотографиям разных людей. Тренажер сразу показывает верно или неверно была угадана эмоция и считает процент верных ответов.

*Micro Expression Trainer.* Это программа для тренировки чтения лиц. Микровыражения выражают семь эмоций (счастье, грусть, гнев, отвращение, презрение, страх и удивление), которые может испытывать человек. Тренажер имеет множество различных персонажей, благодаря чему улучшается распознавание выражений лица.

Сравним рассмотренные аналоги (таблица 2):

Таблица 2– Сравнение аналогов

|  |  |  |
| --- | --- | --- |
| **Название** | **Предназначена для детей дошкольного возраста** | **Распознавание эмоций** |
| 1С:Дошкольная психодиагностика. Базовая версия | Предназначена | Отсутствует |
| Мимика | Не предназначена | Присутствует |
| Micro Expression Trainer | Не предназначена | Присутствует |

Последние два рассмотренные аналога не предназначены для детей дошкольного возраста. «1С:Дошкольная психодиагностика. Базовая версия» содержит довольно большой спектр различных методик и тестов, но не предоставляет возможности выбрать ребенку соответствующую эмоцию в какой-то ситуации.

* 1. **Требования**

*Функциональные:*

Интерактивное приложение должно:

* иметь возможность выбрать аватар;
* давать возможность выбрать набор карточек;
* содержать игру «Детектив», игру «Подбери маску» и игру «Пятнашки»;
* должно выводить статистику верных/неверных;
* иметь звуковое сопровождение

*Нефункциональные:*

* Эстетичность интерфейса интерактивного приложения.
* Удобство, интуитивно понятный интерфейс

Наличие интуитивно-понятного интерфейса

* Совместимость с интерактивным столом SKY 360 [25].

Запуск программы с интерактивного стола

* 1. **Методы решения**

Для решения поставленных задач были выбраны метод объектно-ориентированного анализа и метод моделирования. Реализация первого метода осуществляется с помощью *объектно-ориентированного программирования* (ООП). В отличие от обычно процедурного программирования при ООП основными элементами программы являются не переменные и методы (функции), а объекты.

Объекты – это программные конструкции, включающие набор генетически связанных свойств (данных) и методов [35]. Между объектами происходит взаимодействие по средствам вызова методов. Поля объектов (переменные) фиксируют его состояние, а вызов метода приводит к реакции объекта и (или) изменению его состояния (изменения переменных внутри объекта) [36].

Состояние объекта характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств [39].

При ООП программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов из взаимодействия и операций над объектами (или методов) [14].

В нашем коде классом будет *TApp*. Основной метод *\_\_init\_\_* вызывается при создании объекта на основе этого класса. В качестве атрибута выступает слово *self*, которое ссылается на сам объект (Рисунок 1)

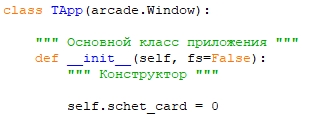


Рисунок 1 – Программам методом ООП

Для решения поставленных задач метод ООП наиболее удобен.

*Метод моделирования* реализуется с помощью карточек эмоций.

Моделирование – метод получения новых знаний путем вывода по аналогии. Модель при этом определяется как некоторая система отношений, присущая другому объекту или абстрактной конструкции [6]. Основной сущностной характеристикой моделирования является его опосредованность, так как в качестве своеобразного инструмента познания выступает объект – заместитель оригинала, модель, исследование которой и позволяет получить новое знание об изучаемом явлении [12]. В нашем случае в качестве модели выступают карточки с изображением какой-либо эмоции.

# **ГЛАВА 2. РЕАЛИЗАЦИЯ ИНТЕРАКТИВНОГО ПРИЛОЖЕНИЯ ДЛЯ ДИАГНОСТИКИ ПСИХОЛОГИЧЕСКИХ ОСОБЕННОСТЕЙ ДЕТЕЙ**

* 1. **Постановка задачи**

Требуется разработать интерактивное приложение для операционной системы Windows, в которой будут учтены все требования к этому продукту.

Приложение должно из файлов загружать картинки, которые представляют различные эмоции. По заданию тестируемый должен выбрать один из вариантов ответа, нажав на соответствующую картинку с эмоцией. При верно указанном ответе программа воспроизводит звук, свидетельствующий о том, что выбранная картинка соответствует правильному ответу на задание. При неверно указанном ответе программа воспроизводит звук, свидетельствующий о том, что выбранная картинка соответствует неправильно выбранному ответу на задание.

Также при запуске какой-то из игры (интерактивное приложение имеет три игры) идет звуковое сопровождение с описанием задания.

* 1. **Выбор и описание средств разработки**

Для того, чтобы выбрать язык программирования для реализации интерактивного приложения, были рассмотрены несколько языков программирования: С#, Python 3.7, Java.

Наиболее известным примером объектно-ориентированного языка программирования является язык C++, развившийся из императивного языка C. Его прямым потомком и логическим продолжением является язык C# [13].

Язык *C#* является самым новым из широко известных языков программирования. Этот язык является строго типизированным, т.е. данные, с которыми он работает, имеют явно или неявно заданный тип. C# – объектно-ориентированный язык, предназначенный для разработки различных видов приложений. [26] Написанная на этом языке программа будет представлять собой совокупность взаимодействующих между собой классов [15]. Язык C# дает возможность программисту вводить свои (пользовательские) типы не только с помощью классов, но и с помощью структур, перечислений, интерфейсов и делегатов. Это кроссплатформенный язык, который вначале создавался с ориентацией на эффективную работу с платформой .NET Framework. В свою очередь, платформа .NET Framework – это набор библиотек. Инфраструктура этой платформы делает C# весьма универсальным средством для разработки портабельных (переносимых) приложений [26].

*Java* – это язык программирования для создания безопасных, переносимых, надежных, объектно-ориентированных интерактивных программ с параллельно выполняющими подпроцессами [34]. Одним из главных свойств Java является поддержка объектно-ориентированного программирования. Объектно-ориентированных подход к программированию позволяет разработать достаточно сложные программы [3].

Java объединяет в себе язык, высококачественную среду выполнения программ и обширную библиотеку. Синтаксис Java представляет собой упрощенный вариант синтаксиса C++ [2]. Технология Java является одновременно и языком программирования, и платформой [19]. Поэтому также Java – это программная платформа, включающая в себя мощную библиотеку, большой объем кода, пригодного для повторного использования, а также среду для выполнения программ, которая обеспечивает безопасность, независимость от операционной системы и автоматическую «сборку мусора» [2].

*Python* – это высокоуровневый язык программирования. Он сочетает в себе мощь языков программирования с простотой использования предметно-ориентированных скриптовых языков. В Python есть библиотеки для загрузки данных, визуализации, статических вычислений, обработки естественного языка, обработки изображений и многого другого. В качестве языка программирования общего назначения Python позволяет создавать сложные графические пользовательские интерфейсы (GUI) и веб-сервисы, а также легко интегрироваться в уже существующие системы[21]. Он оптимизирован для создания качественного программного обеспечения, высокой производительности труда разработчиков, переносимости программ и интеграции компонентов. Python отличается удивительной простотой, удобочитаемостью и простым синтаксисом; он легко интегрируется с внешними компонентами, написанными на других языках программирования; имеет мультипарадигменную архитектуру и поддерживает объектно-ориентированное, функциональное и модульное программирование; обладает обширной коллекцией уже запрограммированных интерфейсов и утилит. Набор встроенных инструментальных средств делает его необычайно гибким и динамическим языком программирования, идеально подходящего не только для быстрого решения тактических задач, но и для разработки перспективных стратегических решений [18].

Языки Java и C# принимают исходный код и компилируют его на машинный язык в виртуальную машину. В итоге получается отдельная программа, которая действует как слой между реальной машиной и скомпилированным кодом. Python является отличным языком для начала программирования. И меньше работы требуется, чтобы сделать графику [4]. Поэтому для разработки интерактивного приложения был выбран высокоуровневый язык программирования Python 3.7 [37].

* 1. **Описание функционала предлагаемого решения**

В таблице 3 приводится сопоставление требований к разрабатываемому продукту и субъектов, работающих в данными требованиями, и прецедентов, реализующие данное требование.

Таблица 3 – Сопоставление требований, субъектов и прецедентов

| № | Требование | Субъект | Прецедент |
| --- | --- | --- | --- |
| Т1 | Интерактивное приложение должно иметь возможность выбора аватара | Тестируемый, тестирующий | Выбор аватара |
| Т2 | Интерактивное приложение должно давать возможность выбрать набор карточек | Тестируемый, тестирующий | Выбор набора карточек |
| Т3 | Интерактивный приложение должно содержать:  • игру «Детектив»;  • игру «Подбери маску»;  • игру «Пятнашки»; | Тестируемый, тестирующий | Выбор игры |
| Т4 | Интерактивное приложение должно выводить статистику верных/неверных | Тестирующий | Просмотр статистики |
| Т5 | Интерактивное приложение должно увеличивать уровень сложности |  | Увеличение уровня сложности |
| Т6 | Интерактивное приложение должно иметь звуковое сопровождение |  | Звуковое сопровождение |

На основе полученной таблицы можно построить диаграмму прецедентов (рисунок 2).

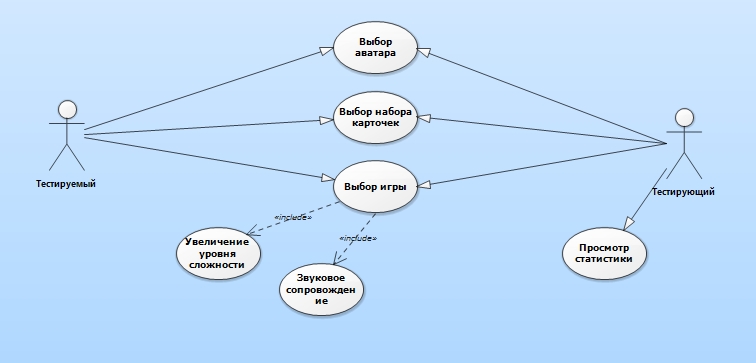


Рисунок 2 – Диаграмма прецедентов

Также можно привести спецификацию прецедентов (таблица 4).

Таблица 4 – Спецификация прецедентов

|  |  |
| --- | --- |
| **Прецедент I**. Выбор аватара | |
| *Краткое описание* | Выбор аватара тестируемым или тестирующим |
| *Субъекты* | Тестируемый, тестирующий |
| *Предусловия* | Прецедент выполняется под учетной записью любого из выбранного аватара |
| *Основной поток* | 1. Прецедент начинается с запуска приложения  2. Тестируемый выбирает аватар  А1. *Аватар не выбран* |
| *Альтернативный поток* | А1. *Аватар не выбран*   1. Интерактивное приложение по умолчанию выбирает первый из представленных аватаров 2. Прецедент заканчивается |
| *Постусловия* |  |
|  | |
| **Прецедент II.** Выбор набора карточек | |
| *Краткое описание* | Выбор набора карточек тестируемым или тестирующим |
| *Субъекты* | Тестируемый, тестирующий |
| *Предусловия* | Прецедент выполняется под учетной записью любого из выбранного аватара |
| *Основной поток* | 1. Прецедент начинается с перехода на выбор набора карточек  2. Тестируемый выбирает набор карточек  А1. *Набор не выбран* |
| *Альтернативный поток* | А1. *Набор не выбран*   1. Интерактивное приложение по умолчанию выбирает первый из представленных наборов 2. Прецедент заканчивается |
| *Постусловия* |  |
|  | |
| **Прецедент III.** Выбор игры | |
| *Краткое описание* | Выбор игра тестируемым или тестирующим |
| *Субъекты* | Тестируемый, тестирующий |
| *Предусловия* | Прецедент выполняется под учетной записью любого из выбранного аватара |
| *Основной поток* | 1. Прецедент начинается с перехода на выбор игры  2. Тестирующий или тестируемый выбирает игру |
| *Альтернативный поток* |  |
| *Постусловия* |  |
|  |  |
| **Прецедент IV**. Просмотр статистики | |
| *Краткое описание* | Просмотр статистики тестирующим |
| *Субъекты* | Тестирующий |
| *Предусловия* | Прецедент выполняется под учетной записью любого из выбранного аватара |
| *Основной поток* | 1. Прецедент начинается с перехода статистику  2. Тестирующий просматривает статистику |
| *Альтернативный поток* |  |
| *Постусловия* |  |
|  | |
| **Прецедент V.** Увеличение уровня сложности | |
| *Краткое описание* | Увеличение уровня сложности игры |
| *Субъекты* | Тестируемый |
| *Предусловия* | Прецедент выполняется под учетной записью любого из выбранного аватара |
| *Основной поток* | 1. Прецедент начинается после запуска игры  2. Тестируемый нажимает на спрайт «Далее», чтобы перейти к следующему уровню сложности |
| *Альтернативный поток* |  |
| *Постусловия* |  |
|  | |
| **Прецедент VI.** Звуковое сопровождение | |
| *Краткое описание* | Звуковое сопровождение игр |
| *Субъекты* | Тестируемый |
| *Предусловия* | Прецедент выполняется под учетной записью любого из выбранного аватара |
| *Основной поток* | 1. Прецедент начинается с запуска игры  2. При выборе верного или неверного ответа программа воспроизводит соответствующий звук |
| *Альтернативный поток* |  |
| *Постусловия* |  |
|  | |

Для каждого из предъявляемого требования мы определили субъекты и прецеденты и привели спецификацию прецедентов.

* 1. **Функциональная модель данных**

Для того, чтобы разработать интерактивное приложение, необходимо определить стрелки функционально модели IDEF0 (рисунок 3).

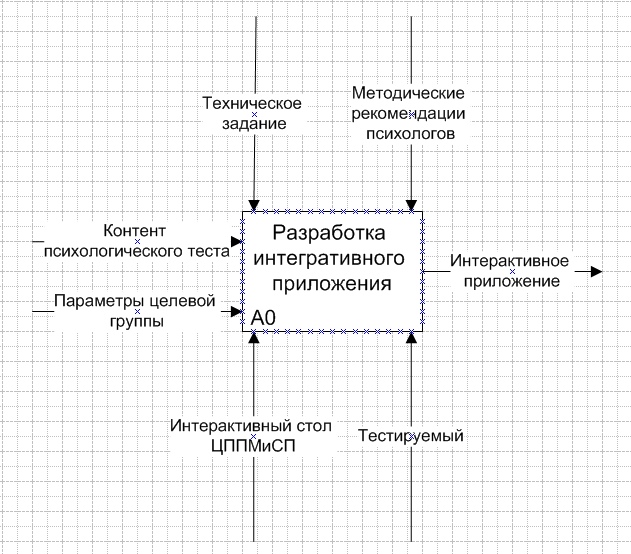


Рисунок 3 – Функциональная модель IDEF0

*Вход* – «Контен психологического теста», «Параметры целевой группы». Это материал или информация, используемые для получения Выхода.

*Управление* – «Техническое задание», «Методические рекромендации психологов». Это управляющие и регулирующие данные, которыми руководствуется функция.

*Выход* – «Интерактивное приложение. Это результат выполнения функции.

*Механизм* – «Интерактивный стол ЦППМиСП», «Тестируемый». Это ресурсы, которые непосредственно реализуют функцию [10].

* 1. **Необходимые для реализации библиотеки и модули**

*Библиотека Arcade* – это библиотека для создания игр. В отличие от широко известной библиотеки Pygame, Arcade имеет ряд особенностей. Поддерживает анимационные спрайты, стационарные спрайты рисуются быстрее. Начало системы координат находится в левом нижнем углу. Улучшенная документация по API для команд и многое другое.

*Модуль OS* – это библиоте6ка, предназначенная для работы с установленной операционной системой и файлами компьютера. Благодаря ей программы, что нужны для работы, не зависят от типа операционной системы. Также их можно легко переносить на другую платформу. В os входит множество методов, которые позволяют создавать и изменять рабочую директорию и удалять из неё файлы; вычислять размер, переименовывать, получать информацию об объекте; обрабатывать пути с помощью класса path.

*Модуль Tkinter* – библиотека, отвечающая за графический интерфейс оконного приложения. Она включает в себя множество виджетов, обладающих набором аргументов.

*Библиотека Random*. С её помощью можно генерировать случайные целые и вещественные числа, буквы, диапазоны вещественных и целых чисел, использовать в генераторах списка, перемешивать и другое.

*Библиотека Pyglet* – это библиотека мультимедиа и окон. Её используют чаще всего для разработки игр и приложений. Pyglet поддерживает загрузку видео, изображений; воспроизведения музыки, звуков;, графику OpenGL и другое.

*Библиотека Typing* – модуль для работы с типами данных.

* 1. **Проектирование пользовательского интерфейса**

Одной из главных частей любого программного продукта является пользовательский интерфейс.

В данном разделе представлены элементы управления и схемы расположения основных элементов состояний проектируемого приложения.

Состояние «Выбор аватара», которое отвечает за авторизацию тестируемого (рисунок 4 и таблица 5).



Рисунок 4 – Состояние «Выбор аватара»

Таблица 5 – Состояние «Выбор аватара»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Выбор аватара» | В верхней части экрана по середине | Размер: 450х65  Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Спрайт «Зайчик» | В средней части экрана в верхней строке слева | Размер: 100х100 | Спрайт выбора аватара «Зайчик» |
| Спрайт «Совенок» | В средней части экрана в верхней строке правее кнопки «Зайчик» | Размер: 100х100 | Спрайт выбора аватара «Совенок» |
| Спрайт «Львенок» | В средней части экрана в верхней строке правее кнопки «Совенок» | Размер: 100х100 | Спрайт выбора аватара «Львенок» |
| Спрайт «Медвежонок» | В средней части экрана в верхней строке справа | Размер: 100х100 | Спрайт выбора аватара «Медвежонок» |
| Спрайт «Котик» | В средней части экрана в нижней строке слева | Размер: 100х100 | Спрайт выбора аватара «Котик» |
| Спрайт «Собачка» | В средней части экрана в нижней строке правее кнопки «Котик» | Размер: 100х100 | Спрайт выбора аватара «Собачка» |
| Спрайт а «Рыбка» | В средней части экрана в нижней строке правее кнопки «Рыбка» | Размер: 100х100 | Спрайт выбора аватара «Рыбка» |
| Спрайт «Слоник» | В средней части экрана в нижней строке слева | Размер: 100х100 | Спрайт выбора аватара «Слоник» |
| Изображение «Стрелка» | В нижней части экрана справа | Размер: 615х346 | Изображение |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Выбор набора карточек», которое отвечает за выбор набора изображений в дальнейшем тестировании (рисунок 5 и таблица 6).

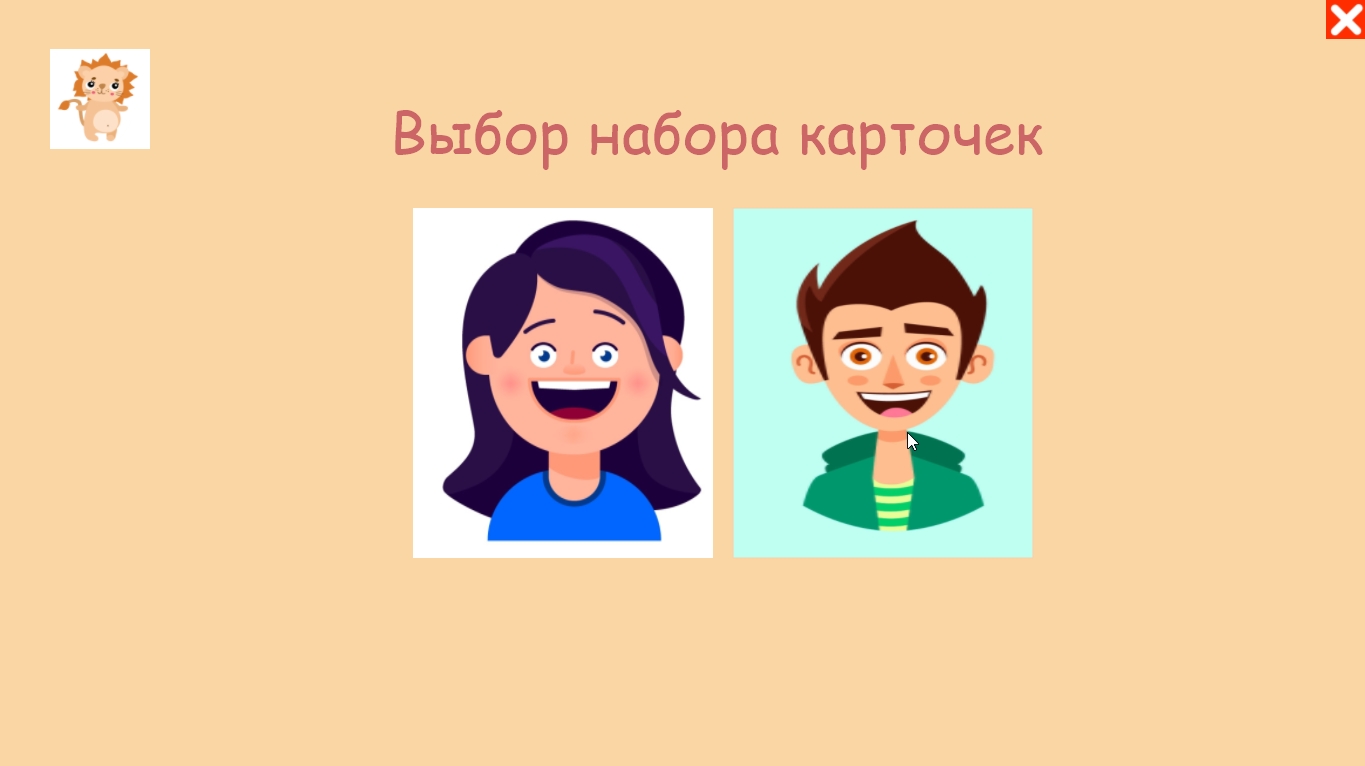


Рисунок 5 – Состояние «Выбор набора карточек»

Таблица 6 – Состояние «Выбор набора карточек»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Выбор набора карточек» | В верхней части экрана по середине | Размер: 550х65  Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Спрайт «Девочка» | В средней части экрана слева | Размер: 300х350 | Спрайт выбора набора карточек «Девочка» |
| Спрайт «Мальчик» | В средней части экрана справа | Размер: 300х350 | Спрайт выбора набора карточек «Мальчик» |
| Изображение «Львенок» | В верхней части экрана слева | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Выбор набора карточек», которое отвечает за выбор набора изображений в дальнейшем тестировании (рисунок 6 и таблица 7).

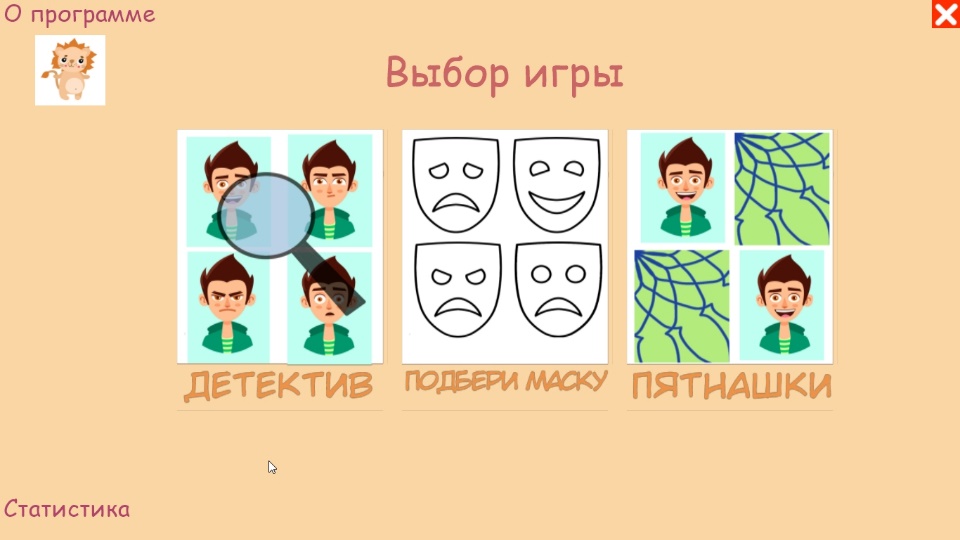


Рисунок 6 – Состояние «Выбор игры»

Таблица 7 – Состояние «Выбор игры»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Выбор игры» | В верхней части экрана слева | Размер: 450х65  Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Спрайт «О программе» | В верхней части экрана слева | Размер: 400х8  Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в состояние «О программе» |
| Спрайт «Статистика» | В нижней части экрана слева | Размер: 400х8  Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в состояние «Статистика» |
| Спрайт «Детектив» | В средней части экрана слева | Размер: 300х400 | Спрайт перехода в состояние «Детектив» |
| Спрайт «Подбери маску» | В средней части экрана по середине | Размер: 300х400 | Спрайт перехода в состояние «Подбери маску» |
| Спрайт «Пятнашки» | В средней части экрана справа | Размер: 300х400 | Спрайт перехода в состояние «Пятнашки» |
| Изображение «Львенок» | В верхней части экрана слева ниже спрайта «О программе» | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «О программе», которое показывает информацию о программе (рисунок 7 и таблица 8).



Рисунок 7 – Состояние «О программе»

Таблица 8 – Состояние «О программе»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «О программе» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Программа предназначена для» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «диагностики психологических особенностей детей» | В центральной части экрана вверху ниже метки «Программа предназначена для» | Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Заказчик:» | В центральной части экрана вверху ниже метки «диагностики психологических особенностей детей» | Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Гобу Мурманской области центр психолого-педагогической,» | В центральной части экрана вверху ниже метки «Заказчик:» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «медицинской и социальной помощи» | В центральной части экрана вверху ниже метки «Гобу Мурманской области центр психолого-педагогической,» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Разработчики:» | В центральной части экрана вверху ниже метки «медицинской и социальной помощи» | Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Дарья Сумина (студентка 4 курса, группа МКН)» | В центральной части экрана вверху ниже метки «Разработчики:» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Олег Иванович Ляш (руководитель)» | В центральной части экрана вверху ниже метки «Дарья Сумина (студентка 4 курса, группа МКН)» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Designed by Freepik» | В центральной части экрана вверху ниже метки «Олег Иванович Ляш (руководитель)» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Изображение «Стрелка» | В нижней части экрана слева | Размер: 615х346 | Изображение |
| Спрайт «Назад» | В нижней части экрана слева | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в предыдущее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Статистика», которое показывает информацию о программе (рисунок 8 и таблица 9).



Рисунок 8 – Состояние «Статистика»

Таблица 9 – Состояние «О программе»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «О программе» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Программа предназначена для» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «диагностики психологических особенностей детей» | В центральной части экрана вверху ниже метки «Программа предназначена для» | Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Заказчик:» | В центральной части экрана вверху ниже метки «диагностики психологических особенностей детей» | Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Гобу Мурманской области центр психолого-педагогической,» | В центральной части экрана вверху ниже метки «Заказчик:» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «медицинской и социальной помощи» | В центральной части экрана вверху ниже метки «Гобу Мурманской области центр психолого-педагогической,» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Разработчики:» | В центральной части экрана вверху ниже метки «медицинской и социальной помощи» | Шрифт: Comic Sans MS, 25 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Дарья Сумина (студентка 4 курса, группа МКН)» | В центральной части экрана вверху ниже метки «Разработчики:» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Олег Иванович Ляш (руководитель)» | В центральной части экрана вверху ниже метки «Дарья Сумина (студентка 4 курса, группа МКН)» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Designed by Freepik» | В центральной части экрана вверху ниже метки «Олег Иванович Ляш (руководитель)» | Шрифт: Comic Sans MS, 25 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Изображение «Стрелка» | В нижней части экрана слева | Размер: 615х346 | Изображение |
| Спрайт «Назад» | В нижней части экрана слева | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в предыдущее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Игра Детектив 1», которое отвечает за певую игру «Детектив» (рисунок 9 и таблица 10).



Рисунок 9 – Состояние «Игра Детектив 1»

Таблица 10 – Состояние «Игра Детектив 1»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Игра Детектив» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Найди мальчика, которому подарили подарок» | В центральной части экрана вверху под меткой «Найди мальчика, которому подарили подарок» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Какое у него настроение?» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Спрайт «Грусть» | В центральной части экрана слева | Размер: 150х200 | Спрайт, обозначающий эмоцию «Грусть» |
| Спрайт «Радость | В центральной части экрана справа | Размер: 150х200 | Спрайт, обозначающий эмоцию «Радость» |
| Изображение «Львенок» | В верхней части экрана слева ниже спрайта «О программе» | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Изображение «Стрелка» | В нижней части экрана справа | Размер: 615х346 | Изображение |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Игра Детектив 2», которое отвечает за вторую игру «Детектив» (рисунок 10 и таблица 11).



Рисунок 10 – Состояние «Игра Детектив 2»

Таблица 11 – Состояние «Игра Детектив 2»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Игра Детектив» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Найти мальчика, который потерял конфету» | В центральной части экрана вверху под меткой «Найди мальчика, которому подарили подарок» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Какое у него настроение?» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Спрайт «Грусть» | В центральной части экрана слева внизу | Размер: 150х200 | Спрайт, обозначающий эмоцию «Грусть» |
| Спрайт «Радость» | В центральной части экрана справа снизу | Размер: 150х200 | Спрайт, обозначающий эмоцию «Радость» |
| Спрайт «Злость» | В центральной части экрана слева вверху | Размер: 150х200 | Спрайт, обозначающий эмоцию «Злость» |
| Спрайт «Страх» | В центральной части экрана справа вверху | Размер: 150х200 | Спрайт, обозначающий эмоцию «Страх» |
| Изображение «Львенок» | В верхней части экрана слева ниже спрайта «О программе» | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Изображение «Стрелка» | В нижней части экрана справа | Размер: 615х346 | Изображение |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Игра Детектив 3», которое отвечает за третью игру «Детектив» (рисунок 10 и таблица 12).



Рисунок 10 – Состояние «Игра Детектив 3»

Таблица 12 – Состояние «Игра Детектив 3»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Игра Детектив» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Найди мальчика, которому подарили подарок» | В центральной части экрана вверху под меткой «Найди мальчика, которому подарили подарок» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Какое у него настроение?» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Спрайт «Удивление» | В центральной части экрана слева внизу | Размер: 150х200 | Спрайт, обозначающий эмоцию «Удивление» |
| Спрайт «Стыд» | В центральной части экрана по середине внизу | Размер: 150х200 | Спрайт, обозначающий эмоцию «Стыд» |
| Спрайт «Страх» | В центральной части экрана справа внизу | Размер: 150х200 | Спрайт, обозначающий эмоцию «Страх» |
| Спрайт «Радость» | В центральной части экрана слева вверху | Размер: 150х200 | Спрайт, обозначающий эмоцию «Радость» |
| Спрайт «Грусть» | В центральной части экрана по середине вверху | Размер: 150х200 | Спрайт, обозначающий эмоцию «Грусть» |
| Спрайт «Злость» | В центральной части экрана справа вверху | Размер: 150х200 | Спрайт, обозначающий эмоцию «Злость» |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Игра Подбери маску 1», которое отвечает за первую игру «Подбери маску» (рисунок 12 и таблица 13).



Рисунок 12 – Состояние «Игра Подбери маску 1»

Таблица 13 – Состояние «Игра Подбери маску 1»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Игра Подбери маску» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Саша очень любит играть на площадке перед домом,» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «но сегодня мама его не пустила. Найди Сашу.» | В центральной части экрана вверху под меткой «Саша очень любит играть на площадке перед домом,» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Какое у него настроение?» | В центральной части экрана вверху под меткой «но сегодня мама его не пустила. Найди Сашу.» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Изображение «Дождь» | В центральной части экрана вверху | Размер: 360х324 | Изображение для задания |
| Спрайт «маска грусти» | В нижней части экрана справа | Размер: 100х100 | Спрайт, обозначающий маску «Грусть» |
| Спрайт «маска радости» | В нижней части экрана слева | Размер: 100х100 | Спрайт, обозначающий маску «Радость» |
| Изображение «Львенок» | В верхней части экрана слева ниже спрайта «О программе» | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Изображение «Стрелка» | В нижней части экрана справа | Размер: 615х346 | Изображение |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Игра Подбери маску 2», которое отвечает за вторую игру «Подбери маску» (рисунок 13 и таблица 14).



Рисунок 13 – Состояние «Игра Подбери маску 2»

Таблица 14 – Состояние «Игра Подбери маску 2»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Игра Подбери маску» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Саша очень любит играть на площадке перед домом,» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «но сегодня мама его не пустила. Найди Сашу.» | В центральной части экрана вверху под меткой «Саша очень любит играть на площадке перед домом,» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Какое у него настроение?» | В центральной части экрана вверху под меткой «но сегодня мама его не пустила. Найди Сашу.» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Изображение «Злость» | В центральной части экрана вверху | Размер: 360х324 | Изображение для задания |
| Спрайт «маска радости» | В нижней части экрана слева | Размер: 100х100 | Спрайт, обозначающий маску «Радость» |
| Спрайт «маска грусти» | В нижней части экрана правее спрайта «маска радости» | Размер: 100х100 | Спрайт, обозначающий маску «Грусти» |
| Спрайт «маска злости» | В нижней части экрана правее спрайта «маска грусти» | Размер: 100х100 | Спрайт, обозначающий маску «Злости» |
| Спрайт «маска страха» | В нижней части экрана слева | Размер: 100х100 | Спрайт, обозначающий маску «Страха» |
| Изображение «Львенок» | В верхней части экрана слева ниже спрайта «О программе» | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Изображение «Стрелка» | В нижней части экрана справа | Размер: 615х346 | Изображение |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Игра Подбери маску 3», которое отвечает за третью игру «Подбери маску» (рисунок 14 и таблица 15).



Рисунок 14 – Состояние «Игра Подбери маску 3»

Таблица 15 – Состояние «Игра Подбери маску 3»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Игра Подбери маску» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Метка «Сёма приехал с родителями в первый раз на море.» | В центральной части экрана вверху | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Найди где Сёма» | В центральной части экрана вверху под меткой «Саша очень любит играть на площадке перед домом,» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Метка «Какое у него настроение?» | В центральной части экрана вверху под меткой «но сегодня мама его не пустила. Найди Сашу.» | Шрифт: Comic Sans MS, 33 pt  Цвет: IRRESISTIBLE | Информационная метка |
| Изображение «Удивление» | В центральной части экрана вверху | Размер: 360х324 | Изображение для задания |
| Спрайт «маска радости» | В нижней части экрана слева | Размер: 100х100 | Спрайт, обозначающий маску «Радость» |
| Спрайт «маска грусти» | В нижней части экрана справа | Размер: 100х100 | Спрайт, обозначающий маску «Грусть» |
| Спрайт «маска злости» | В нижней части экрана слева | Размер: 100х100 | Спрайт, обозначающий маску «Злость» |
| Спрайт «маска страха» | В нижней части экрана справа | Размер: 100х100 | Спрайт, обозначающий маску «Страх» |
| Спрайт «маска удивления» | В нижней части экрана слева | Размер: 100х100 | Спрайт, обозначающий маску «Удивление» |
| Спрайт «маска стыда» | В нижней части экрана слева | Размер: 100х100 | Спрайт, обозначающий маску «Стыд» |
| Изображение «Львенок» | В верхней части экрана слева ниже спрайта «О программе» | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Изображение «Стрелка» | В нижней части экрана справа | Размер: 615х346 | Изображение |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Состояние «Игра Пятнашки», которое отвечает за игру «Пятнашки» (рисунок 15 и таблица 16).

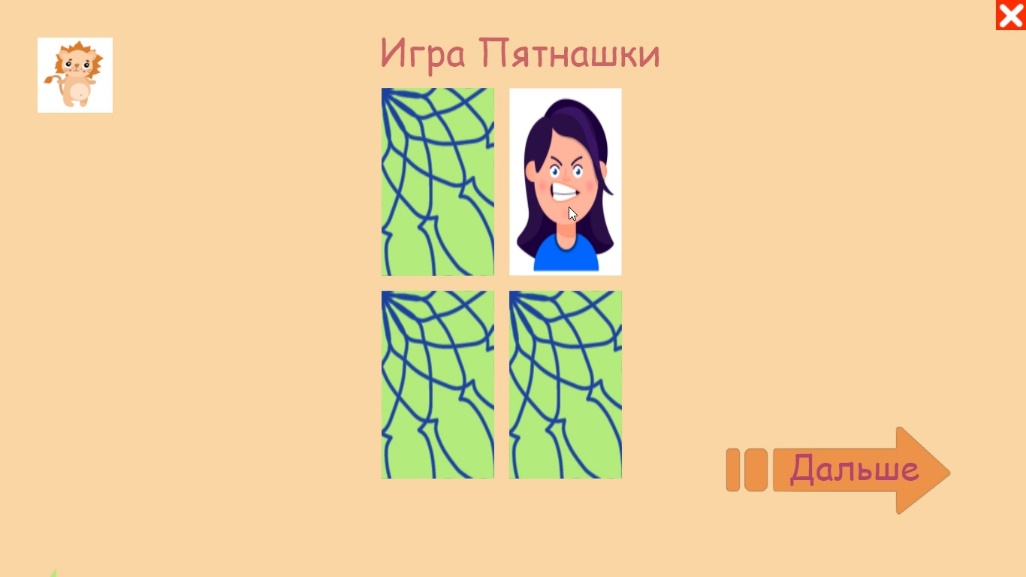


Рисунок 15 – Состояние «Игра Пятнашки»

Таблица 16 – Состояние «Игра Пятнашки»

| **Название элемента** | **Расположение элемента** | **Свойства элемента** | **Назначение элемента** |
| --- | --- | --- | --- |
| Метка «Игра Пятнашки» | В верхней части экрана в центре | Шрифт: Comic Sans MS, 44 pt  Цвет: FUZZY\_WUZZY | Информационная метка |
| Спрайт «Рубашка» | В центральной части экрана вверху справа | Размер: 150х250 | Спрайт, обозначающий «рубашку» карточки |
| Спрайт «Злость» | В центральной части экрана вверху слева | Размер: 150х250 | Спрайт, обозначающий эмоцию «Злость» |
| Спрайт «Рубашка» | В центральной части экрана внизу слева | Размер: 150х250 | Спрайт, обозначающий «рубашку» карточки |
| Спрайт «Рубашка» | В центральной части экрана внизу справа | Размер: 150х250 | Спрайт, обозначающий «рубашку» карточки |
| Изображение «Львенок» | В верхней части экрана слева ниже спрайта «О программе» | Размер: 100х100 | Изображение выбранного аватара «Львенок» |
| Изображение «Стрелка» | В нижней части экрана справа | Размер: 615х346 | Изображение |
| Спрайт «Далее» | В нижней части экрана справа | Размер: 400х15  Шрифт: Comic Sans MS, 35 pt  Цвет: FUZZY\_WUZZY | Спрайт перехода в следующее состояние программы |
| Спрайт «Крестик» | В верхней части экрана справа | Размер: 40х40 | Спрайт выхода из программы |

Интерфейс интерактивного приложения соответствует предъявляемым требованиям. Разработанный продукт готов для диагностики психологических особенностей детей.

# **ЗАКЛЮЧЕНИЕ**

В рамках выпускной квалификационной работы была изучена литература, посвященная теме работы, осуществлен анализ методик психологического тестирования, изучены и проанализированы существующие аналоги, сформулированы требования к собственной разработке, представленные ниже.

Интерактивное приложение должно:

* иметь возможность выбрать аватар;
* давать возможность выбрать набор карточек;
* содержать игру «Детектив», игру «Подбери маску» и игру «Пятнашки»;
* должно выводить статистику верных/неверных;
* иметь звуковое сопровождение

А также выбрано средство разработки, выполнено проектирование программных средств и применены на практике полученные теоретические знания, а именно: реализовано интерактивное приложение для диагностики психологических особенностей детей.

В соответствие с полученными знаниями с помощью языка программирования Python 3.7 и библиотекой Arcade было разработано интерактивное приложение, которое позволяет осуществить диагностику психологических особенностей детей дошкольного возраста. В ходе разработки приложения были получены новые теоретические знания и практические навыки в реализации приложений.

Таким образом, цель работы достигнута, поставленные задачи выполнены в полном объеме.

В перспективе в созданном приложении можно расширить диапазон функционала, добавив возможность тестирующему самому загружать изображения и составлять задание через интерфейс программы.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. The Python Arcade Library [Электронный ресурс] : Режим доступа: <http://arcade.academy/index.html> (Дата обращения: 03.05.2019)
2. Java. Библиотека профессионала. Основы [Текст] : пер. с англ. – М.: ООО «И.Д. Вильямс», 2014. – 864 с. – Т. 1.
3. Java 8: руководство для начинающих [Текст] : пер. с англ. – М.: ООО «И.Д. Вильямс», 2015. – 720 с.
4. Learn Python With Arcade Academy [Электронный ресурс] : Режим доступа: <https://arcade-book.readthedocs.io/en/latest/> (Дата обращения: 20.05.2019)
5. Sweigart, A. Разработка компьютерных игр с помощью Python и Pygame [Текст] / A. Sweigart. – М.: НОУ «Интуит», 2016. – 290 с.
6. Артемьева, О.А. Качественные и количественные методы исследования в психологии [Текст] : учеб. пособие для бакалавриата и магистратуры / О.А. Артемьева. – М.: Изд-во Юрайт, 2019. – 152 с.
7. Белкина, В.Н. Детская психология [Текст] : учеб. пособие для СПО / В.Н. Белкина. – М.: Изд-во Юрайт, 2019. – 170 с.
8. Бизли, Д. Python. Подробный справочник [Текст] / Д. Бизли. – СПб.: Символ-Плюс, 2010. – 84 с.
9. Доунсон, М. Программируем на Python [Текст] / М. Доусон. – СПб.: Питер, 2014. – 416 с.
10. Зараменских, Е.П. Управление жизненным циклом информационных систем [Текст] : учебник и практикум для академического бакалавриата / Е.П. Зараменских. – М.: Изд-во Юрайт, 2019. – 431 с.
11. Златопольский, Д.М. Основы программирования на языке Python [Текст]. / Д. М. Златопольский – М.: ДМК Пресс, 2017. – 284 с.
12. Золотарева, П.В. Методика преподавания по программам дополнительно образования детей [Текст] : учебник и практикум для СПО / П.В. Золотарева, Г. М. Криницкая, А. Л. Пикина. – 2-е изд., испр и доп. – М.: Изд-во Юрайт, 2019. – 315 с.
13. Зыков, С.В. Программирование [Текст] : учебник и практикум для академического бакалавриата / С.В. Зыков. – М.: Изд-во Юрайт, 2019. – 320 с.
14. Зыков, С.В. Программирование. Объектно-ориентированный подход [Текст] : учебник и практикум для академического бакалавриата / С.В. Зыков. – М.: Изд-во Юрайт, 2019. – 155 с.
15. Кудрина, Е.В. Программирование на языке C#: разработка консольных приложений [Текст] / Е.В. Кудрина, М.В. Огнева, М.С. Портенко. – НОУ «Интуит», 2016. – 366 с.
16. Лутц, М. Python. Карманный справочник [Текст] / М. Лутц. – М.: Вильямс, 2015. – 320 с.
17. Лутц, М. Изучаем Python [Текст]. / М. Лутц. – СПб.: Символ-Плюс, 2011. – 1280 с.
18. Лутц, М. Программирование на Python [Текст] : 1 том / М. Лутц. – пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
19. МакГрат, М. Программирование на Java для начинающих [Текст] / М. МакГрат ; [пер. с англ. М. А. Райтмана]. – Москва: Изд-во «Э», 2016. – 192 с.
20. Мэтиз, Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения [Текст] / Э. Мэтиз. – СПб.: Питер, 2017. – 496 с.
21. Миллер, А. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными [Текст] / А. Миллер, С. Гвидо. – М.: Самиздат, 2017. – 393 с.
22. Немов, Р.С. Психология [Текст] : в 2 ч. : учебник для академического бакалавриата / Р.С. Немов. – М.: Изд-во Юрайт, 2019. – 292с. – Ч. 2
23. Немов, Р.С. Психология. Психодиагностика. Введение в научное психологическое исследование с элементами математической статистики [Текст] : в 3 кн. : учеб. для студ. высш. пед. учеб. заведений. / Р.С. Немов. – М.: Гуманитар. изд. центр ВЛАДОС, 2016. – 631 с. – Кн. 3
24. Носс, И.Н. Психодиагностика [Текст]: учебник для академического бакалавриата / И.Н. Носс. – М.: Изд-во Юрайт, 2019. – 500 с.
25. Официальный сайт SKY Interactive [Электронный ресурс] : Режим доступа: <http://sky-interactive.ru/> (Дата обращения: 15.05.2019)
26. Подбельский, В.В. Программирование. Базовый курс C# [Текст] : учебник для СПО / В.В. Подбельский. – Москва: Изд-во Юрайт, 2019. – 369с.
27. Пратик, Дж. Искусственный интеллект с примерами на Python [Текст] / Дж. Пратик. – М. : Вильямс, 2019. – 450 с.
28. Рамальо, Л. Python. К вершинам мастерства [Текст] / Л. Рамальо. – ДМК Пресс, 2016. – 768 с.
29. Раммендик, Д.М. Практикум по психодиагностике [Текст] : учеб. пособие для академического бакалавриата / Д. М. Раммендик, М.Г. Раммендик. – М.: Изд-во Юрайт, 2019. – 193 с.
30. Рейтц, К. Автостопом по Python [Текст] / К. Рейтц, Т. Шлюссер. – СПб.: Питер, 2017. – 336 с.
31. Седер, Н. Python. Экспресс-курс [Текст] / Н. Седер. – СПб.: Питер, 2019. – 479 с.
32. Седжвик, Р. Программирование на языке Python [Текст] / Р. Седжвик, К. Уэйн, Р. Дондеро. – М.: Диалектика 2017. – 736 с.
33. Секреты Python: 59 рекомендаций по написанию эффективного кода [Текст] : пер. с англ. – М. : ООО «И.Д. Вильямс», 2016. – 272 с.
34. Сухов, С.А. Основы программирования на Java [Текст] : учеб. пособие / С.А. Сухов. – Ульяновск: УлГТУ, 2006. – 17 с.
35. Тузовский, А.Ф. Объектно-ориентированное программирование [Текст] : учеб. пособие для прикладного бакалавриата / А.Ф. Тузовский. – М.: Изд-во Юрайт, 2019. – 206 с.
36. Федоров, Д.Ю. Программирование на языке высокого уровня Python [Текст] : учеб. пособие для прикладного бакалавриата / Д.Ю. Федоров. – М.: Изд-во Юрайт, 2019. – 161 с.
37. Федоров, Д.Ю. Программирование на языке высокого уровня Python [Текст] : учеб. пособие для СПО / Д.Ю. Федоров. – М.: Изд-во Юрайт, 2019. – 126 с.
38. Чан Уэсл, Дж. Python: создание приложений [Текст] / Чан Уэсл Дж. – М.: Вильямс, 2015. – 816 с.
39. Черпаков, И.В. Основы программирования [Текст] : учебник и практикум для СПО / И.В. Черпаков. – М.: Изд-во Юрайт, 2019. – 219 с.
40. Шоу, З. Легкий способ выучить Python [Текст] / З. Шоу. – Москва: Эксмо, 2017. – 352 с.

# **ПРИЛОЖЕНИЯ**

**Приложение 1**



**Приложение 2**

1. #-\*- coding:utf-8 -\*-
2. import arcade
3. import os
4. import tkinter as tk
5. import random
6. import pyglet
7. import typing
8. # Включение и отключение вывода отладочных сообщений
9. DEBUG = True
10. class TApp(arcade.Window):
11. """ Основной класс приложения """
12. def \_\_init\_\_(self, fs=False):
13. """ Конструктор """
14. self.schet\_card = 0
15. self.coin=[0,0,0,0]
16. self.otvet=-1
17. self.RAND=[]
18. for j in range(0,2):
19. #print(j)
20. self.RAND.append(j)
21. random.shuffle(self.RAND)
22. #print(self.RAND)
23. self.RAND1=[]
24. for j in range(0,4):
25. #print(j)
26. self.RAND1.append(j)
27. random.shuffle(self.RAND1)
28. #print(self.RAND1)
29. self.RAND2=[]
30. for j in range(0,6):
31. #print(j)
32. self.RAND2.append(j)
33. random.shuffle(self.RAND2)
34. #print(self.RAND2)
35. self.RAND\_1=[]
36. for j in range(0,2):
37. k=random.randint(0,5)
38. for g in range(0,2):
39. self.RAND\_1.append(k)
40. random.shuffle(self.RAND\_1)
41. #print(self.RAND\_1)
42. # Заголовок окна
43. self.title = "Game"
44. self.subtitle = "Диагностика эмоционального развития"
45. # Получаем реальные размеры экрана
46. root = tk.Tk()
47. self.SCREEN\_WIDTH = root.winfo\_screenwidth()
48. #print(root.winfo\_screenwidth())
49. self.SCREEN\_HEIGHT = root.winfo\_screenheight()
50. #print(root.winfo\_screenheight())
51. del root
52. # Параметры масштабирования
53. self.SPRITE\_SCALING = 0.1
54. self.VIEWPORT\_MARGIN = 40
55. # Открываем окно
56. super().\_\_init\_\_(self.SCREEN\_WIDTH, self.SCREEN\_HEIGHT, self.title, fullscreen=True)
57. # Устанавливаем рабочий каталог, где по умолчанию будут находится файлы
58. file\_path = os.path.dirname(os.path.abspath(\_\_file\_\_))
59. os.chdir(file\_path)
60. # Получаем размеры окна и устанавливаем окно просмотра равным этому окну приложения
61. width, height = self.get\_size()
62. #print(width, height)
63. self.set\_viewport(0, width, 0, height)
64. def setup(self):
65. """ Установка основных параметров. """
66. self.setPaths()
67. self.setUserVars()
68. self.setFonts()
69. self.setColors()
70. self.setMenu()
71. self.mouseX = 0
72. self.mouseY = 0
73. self.isMouseDown = False
74. self.loadAvatars()
75. #self.setAbout()
76. def setPaths(self):
77. """ Задаем пути к ресурсам """
78. self.imgPath = "images/"
79. self.logoPath = self.imgPath + "logo/"
80. self.avatarPath = self.imgPath+"avatars/"
81. self.cardPath = self.imgPath+"cards/"
82. self.GUIPath = self.imgPath + "GUI/"
83. self.detectivePath = self.imgPath+"cards/detective/"
84. self.detectivePath1=self.detectivePath+"1/"
85. self.detectivePath2=self.detectivePath+"2/"
86. self.detectivePath3=self.detectivePath+"card/"
87. self.detectivePath4=self.detectivePath+"3/"
88. self.detectivePath5=self.detectivePath+"4/"
89. self.detectivePath6=self.detectivePath+"5/"
90. self.menu1 = self.imgPath+"cards/menu/girl/"
91. self.menu2 = self.imgPath+"cards/menu/boy/"
92. self.soundPath = "sounds/"
93. self.fontPath = "fonts/"
94. self.savePath = "save"
95. self.aboutLogo1 = arcade.Sprite(self.GUIPath + "1.jpg", 0.4)
96. self.aboutLogo3 = arcade.Sprite(self.detectivePath4 + "2.jpg",0.3)
97. self.Fon = arcade.Sprite(self.imgPath + "fon.png", 0.32)
98. self.Strelka = arcade.Sprite(self.imgPath + "Strelka.png", 0.32)
99. self.Strelka1 = arcade.Sprite(self.imgPath + "Strelka2.png", 0.32)
100. def setUserVars(self):
101. """ Переменные описывающие состояние пользователя """
102. # Номер аватара, который выбрал пользователь
103. self.userAvatar = 0
104. #Выбранная карточка
105. self.otvet = -1
106. # Количество правильных ответов
107. self.userGoodAnswers = 0
108. # Количество не правильных ответов
109. self.userBadAnswers = 0
110. #Номер выбранного набора карточек
111. self.userChoiceCards = 0
112. self.MenuItemSelected = 0
113. self.vernotvet = 0
114. def setFonts(self):
115. # шрифты отсюда https://fonts.google.com/?selection.family=Russo+One&subset=cyrillic&sort=popularity
116. self.font\_title="Comic Sans MS"
117. self.font = "Comic Sans MS"
118. def setColors(self):
119. """ Задаем основные цвета """
120. # Цвет фона
121. self.bgcolor = arcade.color.DEEP\_CHAMPAGNE
122. # Цвет текста заголовка
123. self.titlecolor = arcade.color.FUZZY\_WUZZY
124. # Цвет текста подзаголовка
125. self.subtitlecolor = arcade.color.IRRESISTIBLE
126. # Цвет текста пункта меню
127. self.menucolor = arcade.color.IRRESISTIBLE
128. # Цвет текста выбранного пункта меню
129. self.menucolorselected = arcade.color.FRENCH\_LILAC
130. # Задаем фоновый цвет
131. arcade.set\_background\_color(self.bgcolor)
132. #arcade.Texture(1,"fon.png")
133. #arcade.Texture.draw()
134. def setMenu(self):
135. # Переменная состояния приложения
136. # Если = 0, то выводится начальный экран
137. self.state = 0
138. self.state1=99
139. # Словарь для хранения пунктов меню
140. self.Menu = {}
141. # Первый из отображаемых элементов меню
142. self.MenuFirst = 1
143. # Последний из отображаемых пунктов меню
144. self.MenuLast = 5
145. # Собственно сами пункты меню
146. self.Menu[0] = "Стартовое меню"
147. self.Menu[1] = "Выбор набора карточек"
148. self.Menu[2] = "Выбор аватара"
149. self.Menu[3] = "Начать"
150. self.Menu[4] = "О программе"
151. self.Menu[5] = "Выход"
152. self.Menu[6] = "Игра Детектив"
153. self.Menu[7] = "Игра Подбери маску"
154. self.Menu[8] = "Игра Пятнашки"
155. self.Menu[9] = "Дальше"
156. self.Menu[10] = "Игра Детектив"
157. self.Menu[11] = "Игра Детектив"
158. self.Menu[99] = "Пауза"
159. def loadAvatars(self):
160. """ Загрузка ававтаров """
161. files = os.listdir(self.avatarPath)
162. self.imgAvatars = arcade.SpriteList()
163. for i in files:
164. self.imgAvatar = arcade.Sprite(self.avatarPath+i, 1)
165. self.imgAvatar.width = 100
166. self.imgAvatar.height = 100
167. self.imgAvatar.center\_x = 0
168. self.imgAvatar.center\_y = 0
169. self.imgAvatars.append(self.imgAvatar)
170. def on\_draw(self):
171. """ Рендерем экран """
172. arcade.start\_render()
173. if self.state == 0:
174. # Рисуем выбор аватара
175. self.drawState0()
176. elif self.state == 1:
177. # Рисуем выбор набора карточек
178. self.drawState1()
179. elif self.state == 2:
180. # Рисуем выбор игры
181. self.drawState2()
182. elif self.state == 3:
183. # Рисуем игру Детектив 1
184. self.drawState3()
185. elif self.state == 4:
186. # Рисуем игру Подбери маску 1
187. self.drawState4()
188. elif self.state == 5:
189. # Рисуем игру Пятнашки
190. self.drawState5()
191. elif self.state == 6:
192. # Рисуем игру Детектив 2
193. self.drawState6()
194. elif self.state == 7:
195. # Рисуем игру Детектив 3
196. self.drawState7()
197. elif self.state == 8:
198. # Рисуем игру Подбери маску 2
199. self.drawState8()
200. elif self.state == 9:
201. # Рисуем игру Подбери маску 3
202. self.drawState9()
203. elif self.state == 50:
204. # Рисуем О программе
205. self.drawAbout()
206. elif self.state == 51:
207. # Рисуем Статистику
208. self.drawStatistic()
209. elif self.state == 97:
210. #Подсчет верных/неверных ответов
211. self.CountingAnswers()
212. elif self.state == 98:
213. quit()
214. def load\_sound\_library():
215. # lazy loading
216. if not load\_sound\_library.\_sound\_library\_loaded:
217. load\_sound\_library.\_sound\_library\_loaded = True
218. else:
219. return
220. import os
221. appveyor = not os.environ.get('APPVEYOR') is None
222. import platform
223. system = platform.system()
224. if system == 'Windows':
225. import sys
226. is64bit = sys.maxsize > 2\*\*32
227. import site
228. packages = site.getsitepackages()
229. if appveyor:
230. if is64bit:
231. path = "Win64/avbin"
232. else:
233. path = "Win32/avbin"
234. else:
235. if is64bit:
236. path = packages[0] + "/lib/site-packages/arcade/Win64/avbin"
237. else:
238. path = packages[0] + "/lib/site-packages/arcade/Win32/avbin"
239. elif system == 'Darwin':
240. from distutils.sysconfig import get\_python\_lib
241. path = get\_python\_lib() + '/lib/site-packages/arcade/lib/libavbin.10.dylib'
242. pyglet.options['audio'] = ('openal', 'pulse', 'silent')
243. else:
244. path = "avbin"
245. pyglet.options['audio'] = ('openal', 'pulse', 'silent')
246. pyglet.lib.load\_library(path)
247. pyglet.have\_avbin = True
248. # Initialize static function variable
249. load\_sound\_library.\_sound\_library\_loaded = False
250. def load\_sound(filename: str) -> typing.Any:
251. """
252. Load a sound and get it ready to play.
253. """
254. load\_sound\_library()
255. source = pyglet.media.load(filename, streaming=False)
256. return source
257. def play\_sound(sound: typing.Any):
258. """
259. Play a previously loaded sound.
260. """
261. load\_sound\_library()
262. sound.play()
263. def drawState0(self):
264. # Выбор аватара
265. text = "Выбор аватара"
266. color = self.titlecolor
267. text\_size = 44
268. x = self.SCREEN\_WIDTH // 3
269. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 9
270. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
271. self.Fon.center\_x = 0.5 \* self.SCREEN\_WIDTH - 10
272. self.Fon.center\_y = 0.5 \* self.SCREEN\_HEIGHT + 50
273. self.Fon.draw()
274. # Вывод конкретного спрайта
275. w=self.imgAvatars.sprite\_list[1].width
276. h=self.imgAvatars.sprite\_list[1].height
277. counter = 4
278. s = 20
279. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2
280. y = self.SCREEN\_HEIGHT // 2
281. for i in range(0,len(self.imgAvatars.sprite\_list)-1):
282. self.imgAvatars.sprite\_list[i].center\_x = x
283. x += w + s
284. self.imgAvatars.sprite\_list[i].center\_y = y
285. counter -=1
286. if counter <=0:
287. counter = 4
288. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2
289. y += h + s
290. self.imgAvatars.sprite\_list[i].draw()
291. # Определяем попадание курсора на аватар
292. bottom =self.mouseY > self.imgAvatars.sprite\_list[i].center\_y - self.imgAvatars.sprite\_list[i].height // 2
293. top = self.mouseY < self.imgAvatars.sprite\_list[i].center\_y + self.imgAvatars.sprite\_list[i].height // 2
294. left = self.mouseX > self.imgAvatars.sprite\_list[i].center\_x - self.imgAvatars.sprite\_list[i].width // 2
295. right = self.mouseX < self.imgAvatars.sprite\_list[i].center\_x + self.imgAvatars.sprite\_list[i].width // 2
296. if (bottom and top) and (left and right):
297. arcade.draw\_rectangle\_outline(self.imgAvatars.sprite\_list[i].center\_x,self.imgAvatars.sprite\_list[i].center\_y,self.imgAvatars.sprite\_list[i].width,self.imgAvatars.sprite\_list[i].height,color=arcade.color.DEEP\_PEACH)
298. if self.isMouseDown:
299. self.userAvatar = i
300. # Рисуем крестик
301. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
302. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
303. self.aboutLogo1.draw()
304. # Определяем попадание курсора на крестик
305. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
306. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
307. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
308. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
309. if (bottom and top) and (left and right):
310. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.DEEP\_PEACH)
311. if self.isMouseDown:
312. self.MenuItemSelected = 98
313. # Стрелка
314. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.3 + 35
315. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
316. self.Strelka.draw()
317. # Дальше
318. self.MenuFirst2 = 9
319. self.MenuLasti = 1
320. mx = self.mouseX
321. my = self.mouseY
322. width = 400
323. height=15
324. self.MenuItemSelected\_2 = -1
325. for i in range(0,self.MenuLasti):
326. text = self.Menu[self.MenuFirst2+i]
327. text\_size = 35
328. x = self.SCREEN\_WIDTH // 1.3
329. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
330. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
331. color = self.menucolorselected
332. self.MenuItemSelected = 1
333. else:
334. color = self.menucolor
335. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
336. #------------------------------
337. def drawState1(self):
338. # Выбор набора карточек
339. text = "Выбор набора карточек"
340. color = self.titlecolor
341. text\_size = 44
342. x = self.SCREEN\_WIDTH // 3.5
343. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 6
344. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
345. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
346. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
347. self.imgAvatars.sprite\_list[self.userAvatar].draw()
348. """ Загрузка картинок """
349. files1 = os.listdir(self.detectivePath3)
350. self.imgCards1 = arcade.SpriteList()
351. for i in files1:
352. self.imgCard1 = arcade.Sprite(self.detectivePath3+i, 1)
353. self.imgCard1.width = 300
354. self.imgCard1.height = 350
355. self.imgCard1.center\_x = 0
356. self.imgCard1.center\_y = 0
357. self.imgCards1.append(self.imgCard1)
358. # Вывод конкретного спрайта
359. w=self.imgCards1.sprite\_list[1].width
360. h=self.imgCards1.sprite\_list[1].height
361. counter = 2
362. s = 20
363. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 5
364. y = self.SCREEN\_HEIGHT // 2
365. for i in range(0,len(self.imgCards1.sprite\_list)):
366. self.imgCards1.sprite\_list[i].center\_x = x
367. x += w + s
368. self.imgCards1.sprite\_list[i].center\_y = y
369. counter -=1
370. if counter <=0:
371. counter = 5
372. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2.9
373. y += h + s
374. self.imgCards1.sprite\_list[i].draw()
375. # Определяем попадание курсора на набор карточек
376. bottom =self.mouseY > self.imgCards1.sprite\_list[i].center\_y - self.imgCards1.sprite\_list[i].height // 2
377. top = self.mouseY < self.imgCards1.sprite\_list[i].center\_y + self.imgCards1.sprite\_list[i].height // 2
378. left = self.mouseX > self.imgCards1.sprite\_list[i].center\_x - self.imgCards1.sprite\_list[i].width // 2
379. right = self.mouseX < self.imgCards1.sprite\_list[i].center\_x + self.imgCards1.sprite\_list[i].width // 2
380. if (bottom and top) and (left and right):
381. arcade.draw\_rectangle\_outline(self.imgCards1.sprite\_list[i].center\_x,self.imgCards1.sprite\_list[i].center\_y,self.imgCards1.sprite\_list[i].width,self.imgCards1.sprite\_list[i].height,color=arcade.color.DEEP\_PEACH)
382. if self.isMouseDown:
383. self.userChoiceCards=i
384. self.MenuItemSelected = 2
385. #print('выбрали набор карточек ',self.userChoiceCards)
386. # Рисуем крестик
387. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
388. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
389. self.aboutLogo1.draw()
390. # Определяем попадание курсора на крестик
391. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
392. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
393. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
394. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
395. if (bottom and top) and (left and right):
396. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.DEEP\_PEACH)
397. if self.isMouseDown:
398. self.MenuItemSelected = 98
399. def drawState2(self):
400. # Начать
401. text = "Выбор игры"
402. color = self.titlecolor
403. text\_size = 44
404. x = self.SCREEN\_WIDTH // 2.5
405. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 8
406. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
407. self.setMenu1()
408. self.drawMenu1()
409. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
410. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
411. self.imgAvatars.sprite\_list[self.userAvatar].draw()
412. # Рисуем крестик
413. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
414. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
415. self.aboutLogo1.draw()
416. # Определяем попадание курсора на крестик
417. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
418. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
419. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
420. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
421. if (bottom and top) and (left and right):
422. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.DEEP\_PEACH)
423. if self.isMouseDown:
424. self.MenuItemSelected = 98
425. # О программе
426. self.MenuFirst2 = 9
427. self.MenuLasti = 1
428. mx = self.mouseX
429. my = self.mouseY
430. width = 400
431. height=8
432. self.MenuItemSelected\_2 = -1
433. for i in range(0,self.MenuLasti):
434. text = 'О программе'
435. text\_size = 25
436. x = 5
437. y = self.SCREEN\_HEIGHT - 4\* height
438. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
439. color = self.menucolorselected
440. self.MenuItemSelected = 50
441. else:
442. color = self.menucolor
443. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
444. # Статистика
445. self.MenuFirst2 = 9
446. self.MenuLasti = 1
447. mx = self.mouseX
448. my = self.mouseY
449. width = 400
450. height=8
451. self.MenuItemSelected\_2 = -1
452. for i in range(0,self.MenuLasti):
453. text = 'Статистика'
454. text\_size = 25
455. x = 5
456. y = 4\* height
457. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
458. color = self.menucolorselected
459. self.MenuItemSelected = 51
460. else:
461. color = self.menucolor
462. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
463. def drawAbout(self):
464. """ Рисуем о программе """
465. x = self.SCREEN\_WIDTH // 2.5
466. lineHeight = 50
467. # ------------------------
468. text = "О программе"
469. color = self.titlecolor
470. text\_size = 44
471. y = self.SCREEN\_HEIGHT - 2\*lineHeight
472. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
473. text = "Программа предназначена для"
474. color = self.subtitlecolor
475. text\_size = 20
476. x = self.SCREEN\_WIDTH // 2.6
477. y = self.SCREEN\_HEIGHT - 3.3\*lineHeight
478. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font)
479. text = "диагностики психологических особенностей детей"
480. color = self.subtitlecolor
481. text\_size = 20
482. x = self.SCREEN\_WIDTH // 3.4
483. y = self.SCREEN\_HEIGHT - 4\*lineHeight
484. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font)
485. # ------------------------
486. text = "Заказчик:"
487. color = self.titlecolor
488. text\_size = 20
489. x = self.SCREEN\_WIDTH // 2.2
490. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 3
491. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font)
492. text = "Гобу Мурманской области центр психолого-педагогической,"
493. color = self.subtitlecolor
494. text\_size = 20
495. x = self.SCREEN\_WIDTH // 3.9
496. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 3 - 40
497. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font)
498. text = "медицинской и социальной помощи"
499. color = self.subtitlecolor
500. text\_size = 20
501. x = self.SCREEN\_WIDTH // 2.9
502. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 3 - 80
503. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font)
504. #------------------------
505. text = "Разработчики: "
506. color = self.titlecolor
507. text\_size = 20
508. x = self.SCREEN\_WIDTH // 2.3
509. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 2
510. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
511. text = "Дарья Сумина (студентка 4 курса, группа МКН)"
512. color = self.subtitlecolor
513. text\_size = 20
514. x = self.SCREEN\_WIDTH // 3.1
515. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 2 - 40
516. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
517. text = "Олег Иванович Ляш (руководитель)"
518. color = self.subtitlecolor
519. text\_size = 20
520. x = self.SCREEN\_WIDTH // 2.8
521. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 2 - 80
522. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
523. text = "Designed by Freepik"
524. color = self.titlecolor
525. text\_size = 30
526. x = self.SCREEN\_WIDTH // 2.5
527. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 2 - 150
528. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
529. # Стрелка
530. self.Strelka1.center\_x =self.Strelka1.width // 3
531. self.Strelka1.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
532. self.Strelka1.draw()
533. # Назад
534. self.MenuFirst2 = 9
535. self.MenuLasti = 1
536. mx = self.mouseX
537. my = self.mouseY
538. width = 350
539. height=15
540. self.MenuItemSelected\_2 = -1
541. for i in range(0,self.MenuLasti):
542. text = "Назад"
543. text\_size = 35
544. x = self.Strelka1.width//9
545. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
546. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
547. color = self.menucolorselected
548. self.MenuItemSelected = 2
549. else:
550. color = self.menucolor
551. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
552. # Рисуем крестик
553. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
554. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
555. self.aboutLogo1.draw()
556. # Определяем попадание курсора на крестик
557. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
558. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
559. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
560. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
561. if (bottom and top) and (left and right):
562. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.DEEP\_PEACH)
563. if self.isMouseDown:
564. self.MenuItemSelected = 98
565. def drawStatistic(self):
566. # Рисуем статистику
567. text = "Статистика"
568. color = self.titlecolor
569. text\_size = 44
570. x = self.SCREEN\_WIDTH // 2.5
571. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 11
572. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
573. f=open('vern.txt', 'r')
574. Alf=f.readlines()
575. f.close()
576. f=open('nevern.txt', 'r')
577. Alf1=f.readlines()
578. f.close()
579. # Вывод конкретного спрайта для статистики
580. w=self.imgAvatars.sprite\_list[1].width
581. h=self.imgAvatars.sprite\_list[1].height
582. counter = 2
583. s = 450
584. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //0.5
585. y = self.SCREEN\_HEIGHT // 3
586. for i in range(0,len(self.imgAvatars.sprite\_list)-1):
587. self.imgAvatars.sprite\_list[i].center\_x = x
588. x += w + s
589. self.imgAvatars.sprite\_list[i].center\_y = y
590. counter -=1
591. x1 = self.SCREEN\_WIDTH // 2 - (counter \* w) // 0.4
592. x2 = self.SCREEN\_WIDTH // 2 - (counter \* w) //2.7
593. if counter <=0:
594. counter = 2
595. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //0.5
596. x1 = self.SCREEN\_WIDTH // 2 + (counter \* w) // 0.6
597. x2 = self.SCREEN\_WIDTH // 2 + (counter \* w) //0.4
598. # Выводим верные ответы во втором столбце
599. text = "%s" % Alf[i]
600. color = self.titlecolor
601. text\_size = 20
602. arcade.draw\_text(text, x1, y, color, text\_size, anchor\_y="center", font\_name=self.font)
603. # Выводим неверные ответы во втором столбце
604. text1 = "%s" % Alf1[i]
605. color = self.titlecolor
606. text\_size = 20
607. arcade.draw\_text(text1, x2+40, y, color, text\_size, anchor\_y="center", font\_name=self.font)
608. y += h
609. else:
610. # Выводим верные ответы в первом столбце
611. text = "%s" % Alf[i]
612. color = self.titlecolor
613. text\_size = 20
614. arcade.draw\_text(text, x1, y, color, text\_size, anchor\_y="center", font\_name=self.font)
615. # Вывоим неверные ответы в первом столбце
616. text1 = "%s" % Alf1[i]
617. color = self.titlecolor
618. text\_size = 20
619. arcade.draw\_text(text1, x2, y, color, text\_size, anchor\_y="center", font\_name=self.font)
620. self.imgAvatars.sprite\_list[i].draw()
621. s = 500
622. text = "Верные"
623. w=self.imgAvatars.sprite\_list[1].width
624. h=self.imgAvatars.sprite\_list[1].height
625. color = self.titlecolor
626. text\_size = 20
627. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //0.7
628. y = self.SCREEN\_HEIGHT // 1.2
629. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
630. text = "Неверные"
631. w=self.imgAvatars.sprite\_list[1].width
632. h=self.imgAvatars.sprite\_list[1].height
633. color = self.titlecolor
634. text\_size = 20
635. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2.7
636. y = self.SCREEN\_HEIGHT // 1.2
637. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
638. text = "Верные"
639. w=self.imgAvatars.sprite\_list[1].width
640. h=self.imgAvatars.sprite\_list[1].height
641. color = self.titlecolor
642. text\_size = 20
643. x = self.SCREEN\_WIDTH // 2 + (counter \* w) //0.7
644. y = self.SCREEN\_HEIGHT // 1.2
645. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
646. text = "Неверные"
647. w=self.imgAvatars.sprite\_list[1].width
648. h=self.imgAvatars.sprite\_list[1].height
649. color = self.titlecolor
650. text\_size = 20
651. x = self.SCREEN\_WIDTH // 2 + (counter \* w) //0.4
652. y = self.SCREEN\_HEIGHT // 1.2
653. arcade.draw\_text(text, x, y, color, text\_size, anchor\_y="center", font\_name=self.font)
654. # Стрелка
655. self.Strelka1.center\_x =self.Strelka1.width // 3
656. self.Strelka1.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
657. self.Strelka1.draw()
658. # Назад
659. self.MenuFirst2 = 9
660. self.MenuLasti = 1
661. mx = self.mouseX
662. my = self.mouseY
663. width = 400
664. height=15
665. self.MenuItemSelected\_2 = -1
666. for i in range(0,self.MenuLasti):
667. text = "Назад"
668. text\_size = 35
669. x = self.Strelka1.width//9
670. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
671. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
672. color = self.menucolorselected
673. self.MenuItemSelected = 2
674. else:
675. color = self.menucolor
676. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
677. # Рисуем крестик
678. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
679. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
680. self.aboutLogo1.draw()
681. # Определяем попадание курсора на крестик
682. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
683. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
684. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
685. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
686. if (bottom and top) and (left and right):
687. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.DEEP\_PEACH)
688. if self.isMouseDown:
689. self.MenuItemSelected = 98
690. def setMenu1(self):
691. #self.Menu1 = {}
692. # Первый из отображаемых элементов меню
693. self.MenuFirst1 = 6
694. # Последний из отображаемых пунктов меню
695. self.MenuLastii = 3
696. def drawMenu1(self):
697. """ Рисуем менюшку игры """
698. mx = self.mouseX
699. my = self.mouseY
700. width = 400
701. height=15
702. self.MenuItemSelected\_1 = -1
703. """ Загрузка картинок """
704. if self.userChoiceCards==0:
705. files1 = os.listdir(self.menu1)
706. elif self.userChoiceCards == 1:
707. files1 = os.listdir(self.menu2)
708. self.imgMenus = arcade.SpriteList()
709. for i in files1:
710. if self.userChoiceCards==0:
711. self.imgMenu = arcade.Sprite(self.menu1+i, 1)
712. elif self.userChoiceCards==1:
713. self.imgMenu = arcade.Sprite(self.menu2+i, 1)
714. self.imgMenu.width = 300
715. self.imgMenu.height = 400
716. self.imgMenu.center\_x = 0
717. self.imgMenu.center\_y = 0
718. self.imgMenus.append(self.imgMenu)
719. # Вывод конкретного спрайта
720. w=self.imgMenus.sprite\_list[0].width
721. h=self.imgMenus.sprite\_list[0].height
722. counter = 3
723. s = 20
724. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 3.2
725. y = self.SCREEN\_HEIGHT // 2
726. for i in range(0,len(self.imgMenus.sprite\_list)):
727. self.imgMenus.sprite\_list[i].center\_x = x
728. x += w + s
729. self.imgMenus.sprite\_list[i].center\_y = y
730. counter -=1
731. if counter <=0:
732. counter = 3
733. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2.9
734. y += h + s
735. self.imgMenus.sprite\_list[i].draw()
736. # Определяем попадание курсора на набор карточек
737. bottom =self.mouseY > self.imgMenus.sprite\_list[i].center\_y - self.imgMenus.sprite\_list[i].height // 2
738. top = self.mouseY < self.imgMenus.sprite\_list[i].center\_y + self.imgMenus.sprite\_list[i].height // 2
739. left = self.mouseX > self.imgMenus.sprite\_list[i].center\_x - self.imgMenus.sprite\_list[i].width // 2
740. right = self.mouseX < self.imgMenus.sprite\_list[i].center\_x + self.imgMenus.sprite\_list[i].width // 2
741. if (bottom and top) and (left and right):
742. arcade.draw\_rectangle\_outline(self.imgMenus.sprite\_list[i].center\_x,self.imgMenus.sprite\_list[i].center\_y,self.imgMenus.sprite\_list[i].width,self.imgMenus.sprite\_list[i].height,color=arcade.color.DEEP\_PEACH)
743. if self.isMouseDown:
744. self.MenuItemSelected = self.MenuFirst1+i-3
745. self.odin=0
746. self.dva=0
747. self.tri=0
748. # Рисуем крестик
749. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
750. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
751. self.aboutLogo1.draw()
752. # Определяем попадание курсора на крестик
753. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
754. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
755. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
756. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
757. if (bottom and top) and (left and right):
758. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.DEEP\_PEACH)
759. if self.isMouseDown:
760. self.MenuItemSelected = 98
761. def drawState3(self):
762. self.vernotvet=0
763. # Игра Детектив 1
764. self.text="Детектив 1"
765. text = "Игра Детектив"
766. color = self.titlecolor
767. text\_size = 38
768. x = self.SCREEN\_WIDTH // 2.7
769. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 12
770. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
771. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
772. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
773. self.imgAvatars.sprite\_list[self.userAvatar].draw()
774. #print(self.userChoiceCards)
775. if self.userChoiceCards==0:
776. text = "Найти девочку, которой подарили подарок"
777. text2 = "Какое у неё настроение?"
778. files1 = os.listdir(self.detectivePath1)
779. if self.odin==0:
780. self.StateSound()
781. elif self.userChoiceCards == 1:
782. text = "Найти мальчика, которому подарили подарок"
783. text2 = "Какое у него настроение?"
784. files1 = os.listdir(self.detectivePath2)
785. if self.odin==0:
786. self.StateSound()
787. color = self.subtitlecolor
788. text\_size = 33
789. x = self.SCREEN\_WIDTH // 4.2
790. x2 = self.SCREEN\_WIDTH // 3.5
791. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 6
792. y2 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 4
793. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
794. arcade.draw\_text(text2,x2, y2,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
795. #------------------------------
796. """ Загрузка картинок """
797. #files1 = os.listdir(self.detectivePath2)
798. self.imgDetectives1 = arcade.SpriteList()
799. for i in files1:
800. #k=k+1
801. if self.userChoiceCards==0:
802. self.imgDetective1 = arcade.Sprite(self.detectivePath1+i, 1)
803. elif self.userChoiceCards == 1:
804. self.imgDetective1 = arcade.Sprite(self.detectivePath2+i, 1)
805. else:
806. self.imgDetective1 = arcade.Sprite(self.detectivePath1+i, 1)
807. self.imgDetective1.width = 150
808. self.imgDetective1.height = 200
809. self.imgDetective1.center\_x = 0
810. self.imgDetective1.center\_y = 0
811. self.imgDetectives1.append(self.imgDetective1)
812. #-------------
813. # Стрелка
814. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.3 + 35
815. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
816. self.Strelka.draw()
817. # Дальше
818. self.MenuFirst2 = 9
819. self.MenuLasti = 1
820. mx = self.mouseX
821. my = self.mouseY
822. width = 400
823. height=15
824. self.MenuItemSelected\_2 = -1
825. for i in range(0,self.MenuLasti):
826. text = self.Menu[self.MenuFirst2+i]
827. text\_size = 35
828. x = self.SCREEN\_WIDTH // 1.3
829. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
830. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
831. color = self.menucolorselected
832. self.sl=6
833. self.MenuItemSelected = 97
834. else:
835. color = self.menucolor
836. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
837. #------------------------------
838. # Вывод конкретного спрайта
839. w=self.imgDetectives1.sprite\_list[1].width
840. h=self.imgDetectives1.sprite\_list[1].height
841. counter = 2
842. s = 20
843. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2.3
844. y = self.SCREEN\_HEIGHT // 2
845. for i in self.RAND:
846. #print(i)
847. self.imgDetectives1.sprite\_list[i].center\_x = x
848. x += w + s
849. self.imgDetectives1.sprite\_list[i].center\_y = y
850. counter -=1
851. if counter <=0:
852. counter = 2
853. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2.3
854. y += h + s
855. self.imgDetectives1.sprite\_list[i].draw()
856. # Определяем попадание курсора на аватар
857. bottom =self.mouseY > self.imgDetectives1.sprite\_list[i].center\_y - self.imgDetectives1.sprite\_list[i].height // 2
858. top = self.mouseY < self.imgDetectives1.sprite\_list[i].center\_y + self.imgDetectives1.sprite\_list[i].height // 2
859. left = self.mouseX > self.imgDetectives1.sprite\_list[i].center\_x - self.imgDetectives1.sprite\_list[i].width // 2
860. right = self.mouseX < self.imgDetectives1.sprite\_list[i].center\_x + self.imgDetectives1.sprite\_list[i].width // 2
861. if (bottom and top) and (left and right):
862. arcade.draw\_rectangle\_outline(self.imgDetectives1.sprite\_list[i].center\_x,self.imgDetectives1.sprite\_list[i].center\_y,self.imgDetectives1.sprite\_list[i].width,self.imgDetectives1.sprite\_list[i].height,color=arcade.color.DEEP\_PEACH)
863. if self.isMouseDown:
864. self.otvet=i
865. if (self.otvet==self.vernotvet) and (self.tri\_one==0):
866. self.StateSound()
867. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
868. self.StateSound()
869. # Рисуем крестик
870. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
871. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
872. self.aboutLogo1.draw()
873. # Определяем попадание курсора на крестик
874. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
875. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
876. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
877. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
878. if (bottom and top) and (left and right):
879. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.DEEP\_PEACH)
880. if self.isMouseDown:
881. self.MenuItemSelected = 98
882. def StateSound(self):
883. if self.text=="Детектив 1":
884. if (self.userChoiceCards==0) and (self.odin==0):
885. arcade.play\_sound('music/Детектив 1 0.wav')
886. self.odin=1
887. self.state == 3
888. elif (self.userChoiceCards==1) and (self.odin==0):
889. arcade.play\_sound('music/Детектив 1 1.wav')
890. self.odin=1
891. self.state == 3
892. elif (self.otvet==self.vernotvet) and (self.tri\_one==0):
893. arcade.play\_sound('music/Детектив 3 3.wav')
894. self.tri\_one=1
895. self.state == 3
896. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
897. arcade.play\_sound('music/Детектив 3 4.wav')
898. self.tri\_two=1
899. self.state == 3
900. elif self.text=="Детектив 2":
901. if (self.userChoiceCards==0) and (self.dva==0):
902. arcade.play\_sound('music/Детектив 2 0.wav')
903. self.dva=1
904. self.state == 6
905. elif (self.userChoiceCards==1) and (self.dva==0):
906. arcade.play\_sound('music/Детектив 2 1.wav')
907. self.dva=1
908. self.state == 6
909. elif (self.otvet==self.vernotvet) and (self.tri\_one==0):
910. arcade.play\_sound('music/Детектив 3 3.wav')
911. self.tri\_one=1
912. self.state == 6
913. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
914. arcade.play\_sound('music/Детектив 3 4.wav')
915. self.tri\_two=1
916. self.state == 6
917. elif self.text=="Детектив 3":
918. if (self.userChoiceCards==0) and (self.tri==0):
919. arcade.play\_sound('music/Детектив 2 0.wav')
920. self.tri=1
921. self.state == 7
922. elif (self.userChoiceCards==1) and (self.tri==0):
923. arcade.play\_sound('music/Детектив 2 1.wav')
924. self.tri=1
925. self.state == 7
926. elif (self.otvet==self.vernotvet) and (self.tri\_one==0):
927. arcade.play\_sound('music/Детектив 3 3.wav')
928. self.tri\_one=1
929. self.state == 7
930. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
931. arcade.play\_sound('music/Детектив 3 4.wav')
932. self.tri\_two=1
933. self.state == 7
934. elif self.text=="Подбери маску 1":
935. if (self.userChoiceCards==0) and (self.odin==0):
936. arcade.play\_sound('music/Подбери маску 1 0.wav')
937. self.odin=1
938. self.state == 4
939. elif (self.userChoiceCards==1) and (self.odin==0):
940. arcade.play\_sound('music/Подбери маску 1 1.wav')
941. self.odin=1
942. self.state == 4
943. elif (self.otvet==self.vernotvet) and (self.tri\_one==0):
944. arcade.play\_sound('music/Детектив 3 3.wav')
945. self.tri\_one=1
946. self.state == 4
947. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
948. arcade.play\_sound('music/Детектив 3 4.wav')
949. self.tri\_two=1
950. self.state == 4
951. elif self.text=="Подбери маску 2":
952. if (self.userChoiceCards==0) and (self.dva==0):
953. arcade.play\_sound('music/Подбери маску 2 0.wav')
954. self.dva=1
955. self.state == 8
956. elif (self.userChoiceCards==1) and (self.dva==0):
957. arcade.play\_sound('music/Подбери маску 2 1.wav')
958. self.dva=1
959. self.state == 8
960. elif (self.otvet==self.vernotvet) and (self.tri\_one==0):
961. arcade.play\_sound('music/Детектив 3 3.wav')
962. self.tri\_one=1
963. self.state == 8
964. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
965. arcade.play\_sound('music/Детектив 3 4.wav')
966. self.tri\_two=1
967. self.state == 8
968. elif self.text=="Подбери маску 3":
969. if (self.userChoiceCards==0) and (self.tri==0):
970. arcade.play\_sound('music/Детектив 2 0.wav')
971. self.tri=1
972. self.state == 9
973. elif (self.userChoiceCards==1) and (self.tri==0):
974. arcade.play\_sound('music/Детектив 2 1.wav')
975. self.tri=1
976. self.state == 9
977. elif (self.otvet==self.vernotvet) and (self.tri\_one==0):
978. arcade.play\_sound('music/Детектив 3 3.wav')
979. self.tri\_one=1
980. self.state == 9
981. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
982. arcade.play\_sound('music/Детектив 3 4.wav')
983. self.tri\_two=1
984. self.state == 9
985. def drawState6(self):
986. self.vernotvet=1
987. otvet=-1
988. # Игра Детектив 2
989. self.text="Детектив 2"
990. text = "Игра Детектив"
991. color = self.titlecolor
992. text\_size = 38
993. x = self.SCREEN\_WIDTH // 2.7
994. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 12
995. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
996. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
997. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
998. self.imgAvatars.sprite\_list[self.userAvatar].draw()
999. if self.userChoiceCards==0:
1000. text = "Найти девочку, которая потеряла конфету"
1001. text2 = "Какое у неё настроение?"
1002. files1 = os.listdir(self.detectivePath1)
1003. if self.dva==0:
1004. self.StateSound()
1005. elif self.userChoiceCards == 1:
1006. text = "Найти мальчика, который потерял конфету"
1007. text2 = "Какое у него настроение?"
1008. files1 = os.listdir(self.detectivePath2)
1009. if self.dva==0:
1010. self.StateSound()
1011. color = self.subtitlecolor
1012. text\_size = 33
1013. x = self.SCREEN\_WIDTH // 4.2
1014. x2 = self.SCREEN\_WIDTH // 3.5
1015. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 6
1016. y2 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 4
1017. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1018. arcade.draw\_text(text2,x2, y2,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1019. #------------------------------
1020. """ Загрузка картинок """
1021. self.imgDetectives2 = arcade.SpriteList()
1022. for i in files1:
1023. if self.userChoiceCards==0:
1024. self.imgDetective2 = arcade.Sprite(self.detectivePath1+i, 1)
1025. elif self.userChoiceCards == 1:
1026. self.imgDetective2 = arcade.Sprite(self.detectivePath2+i, 1)
1027. else:
1028. self.imgDetective2 = arcade.Sprite(self.detectivePath1+i, 1)
1029. self.imgDetective2.width = 150
1030. self.imgDetective2.height = 200
1031. self.imgDetective2.center\_x = 0
1032. self.imgDetective2.center\_y = 0
1033. self.imgDetectives2.append(self.imgDetective2)
1034. #------------------------------
1035. # Вывод конкретного спрайта
1036. w=self.imgDetectives2.sprite\_list[1].width
1037. h=self.imgDetectives2.sprite\_list[1].height
1038. counter = 2
1039. s = 20
1040. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2.3
1041. y = self.SCREEN\_HEIGHT // 4
1042. for i in self.RAND1:
1043. self.imgDetectives2.sprite\_list[i].center\_x = x
1044. x += w + s
1045. self.imgDetectives2.sprite\_list[i].center\_y = y
1046. counter -=1
1047. if counter <=0:
1048. counter = 2
1049. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2.3
1050. y += h + s
1051. self.imgDetectives2.sprite\_list[i].draw()
1052. # Определяем попадание курсора на аватар
1053. bottom =self.mouseY > self.imgDetectives2.sprite\_list[i].center\_y - self.imgDetectives2.sprite\_list[i].height // 2
1054. top = self.mouseY < self.imgDetectives2.sprite\_list[i].center\_y + self.imgDetectives2.sprite\_list[i].height // 2
1055. left = self.mouseX > self.imgDetectives2.sprite\_list[i].center\_x - self.imgDetectives2.sprite\_list[i].width // 2
1056. right = self.mouseX < self.imgDetectives2.sprite\_list[i].center\_x + self.imgDetectives2.sprite\_list[i].width // 2
1057. if (bottom and top) and (left and right):
1058. arcade.draw\_rectangle\_outline(self.imgDetectives2.sprite\_list[i].center\_x,self.imgDetectives2.sprite\_list[i].center\_y,self.imgDetectives2.sprite\_list[i].width,self.imgDetectives2.sprite\_list[i].height,color=arcade.color.FELDSPAR)
1059. if self.isMouseDown:
1060. self.otvet=i
1061. if (self.otvet==self.vernotvet) and (self.tri\_one==0):
1062. self.StateSound()
1063. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
1064. self.StateSound()
1065. #-------------
1066. # Стрелка
1067. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.3 + 35
1068. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1069. self.Strelka.draw()
1070. # Дальше
1071. self.MenuFirst2 = 9
1072. self.MenuLasti = 1
1073. mx = self.mouseX
1074. my = self.mouseY
1075. width = 400
1076. height=15
1077. self.MenuItemSelected = 6
1078. for i in range(0,self.MenuLasti):
1079. text = self.Menu[self.MenuFirst2+i]
1080. text\_size = 35
1081. x = self.SCREEN\_WIDTH // 1.3
1082. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1083. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
1084. color = self.menucolorselected
1085. self.sl=7
1086. self.MenuItemSelected = 97
1087. else:
1088. color = self.menucolor
1089. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
1090. # Рисуем крестик
1091. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
1092. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
1093. self.aboutLogo1.draw()
1094. # Определяем попадание курсора на крестик
1095. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
1096. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
1097. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
1098. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
1099. if (bottom and top) and (left and right):
1100. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.RED)
1101. if self.isMouseDown:
1102. self.MenuItemSelected = 98
1103. def drawState7(self):
1104. self.vernotvet=3
1105. otvet=-1
1106. # Игра Детектив 3
1107. text = "Игра Детектив"
1108. self.text = "Детектив 3"
1109. color = self.titlecolor
1110. text\_size = 38
1111. x = self.SCREEN\_WIDTH // 2.7
1112. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 12
1113. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1114. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
1115. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
1116. self.imgAvatars.sprite\_list[self.userAvatar].draw()
1117. if self.userChoiceCards==0:
1118. text = "Найти девочку, которая увидела слона"
1119. text2 = "Какое у неё настроение?"
1120. files1 = os.listdir(self.detectivePath1)
1121. if self.tri==0:
1122. self.StateSound()
1123. elif self.userChoiceCards == 1:
1124. text = "Найти мальчика, который увидел слона"
1125. text2 = "Какое у него настроение?"
1126. files1 = os.listdir(self.detectivePath2)
1127. if self.tri==0:
1128. self.StateSound()
1129. color = self.subtitlecolor
1130. text\_size = 33
1131. x = self.SCREEN\_WIDTH // 4.2
1132. x2 = self.SCREEN\_WIDTH // 3.5
1133. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 6
1134. y2 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 4
1135. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1136. arcade.draw\_text(text2,x2, y2,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1137. #------------------------------
1138. """ Загрузка картинок """
1139. self.imgDetectives2 = arcade.SpriteList()
1140. for i in files1:
1141. if self.userChoiceCards==0:
1142. self.imgDetective2 = arcade.Sprite(self.detectivePath1+i, 1)
1143. elif self.userChoiceCards == 1:
1144. self.imgDetective2 = arcade.Sprite(self.detectivePath2+i, 1)
1145. else:
1146. self.imgDetective2 = arcade.Sprite(self.detectivePath1+i, 1)
1147. self.imgDetective2.width = 150
1148. self.imgDetective2.height = 200
1149. self.imgDetective2.center\_x = 0
1150. self.imgDetective2.center\_y = 0
1151. self.imgDetectives2.append(self.imgDetective2)
1152. #-------------
1153. # Стрелка
1154. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.3 + 35
1155. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1156. self.Strelka.draw()
1157. # Дальше
1158. self.MenuFirst2 = 9
1159. self.MenuLasti = 1
1160. mx = self.mouseX
1161. my = self.mouseY
1162. width = 400
1163. height=15
1164. self.MenuItemSelected = 7
1165. for i in range(0,self.MenuLasti):
1166. text = self.Menu[self.MenuFirst2+i]
1167. text\_size = 35
1168. x = self.SCREEN\_WIDTH // 1.3
1169. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1170. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
1171. color = self.menucolorselected
1172. self.sl=2
1173. self.MenuItemSelected = 97
1174. else:
1175. color = self.menucolor
1176. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
1177. #------------------------------
1178. # Вывод конкретного спрайта
1179. w=self.imgDetectives1.sprite\_list[1].width
1180. h=self.imgDetectives1.sprite\_list[1].height
1181. counter = 3
1182. s = 20
1183. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2.3
1184. y = self.SCREEN\_HEIGHT // 4
1185. for i in self.RAND2:
1186. self.imgDetectives1.sprite\_list[i].center\_x = x
1187. x += w + s
1188. self.imgDetectives1.sprite\_list[i].center\_y = y
1189. counter -=1
1190. if counter <=0:
1191. counter = 3
1192. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2.3
1193. y += h + s
1194. self.imgDetectives1.sprite\_list[i].draw()
1195. # Определяем попадание курсора на аватар
1196. bottom =self.mouseY > self.imgDetectives1.sprite\_list[i].center\_y - self.imgDetectives1.sprite\_list[i].height // 2
1197. top = self.mouseY < self.imgDetectives1.sprite\_list[i].center\_y + self.imgDetectives1.sprite\_list[i].height // 2
1198. left = self.mouseX > self.imgDetectives1.sprite\_list[i].center\_x - self.imgDetectives1.sprite\_list[i].width // 2
1199. right = self.mouseX < self.imgDetectives1.sprite\_list[i].center\_x + self.imgDetectives1.sprite\_list[i].width // 2
1200. if (bottom and top) and (left and right):
1201. arcade.draw\_rectangle\_outline(self.imgDetectives1.sprite\_list[i].center\_x,self.imgDetectives1.sprite\_list[i].center\_y,self.imgDetectives1.sprite\_list[i].width,self.imgDetectives1.sprite\_list[i].height,color=arcade.color.FELDSPAR)
1202. if self.isMouseDown:
1203. self.otvet=i
1204. if (self.otvet==self.vernotvet) and (self.tri\_one==0):
1205. self.StateSound()
1206. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
1207. self.StateSound()
1208. # Рисуем крестик
1209. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
1210. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
1211. self.aboutLogo1.draw()
1212. # Определяем попадание курсора на крестик
1213. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
1214. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
1215. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
1216. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
1217. if (bottom and top) and (left and right):
1218. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.FELDSPAR)
1219. if self.isMouseDown:
1220. self.MenuItemSelected = 98
1221. def drawState4(self):
1222. self.vernotvet=1
1223. # Игра Подбери маску
1224. text = "Игра Подбери маску"
1225. self.text = "Подбери маску 1"
1226. color = self.titlecolor
1227. text\_size = 38
1228. x = self.SCREEN\_WIDTH // 2.7
1229. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 12
1230. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1231. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
1232. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
1233. self.imgAvatars.sprite\_list[self.userAvatar].draw()
1234. if self.userChoiceCards==0:
1235. self.aboutLogo2 = arcade.Sprite(self.detectivePath4 + "1\_1.png",0.8)
1236. text = "Саша очень любит играть на плошадке перед домом,"
1237. text2 = "но сегодня мама её не пустила. Найди Сашу."
1238. text3 = "Какое у неё настроение?"
1239. #files1 = os.listdir(self.detectivePath4)
1240. if self.odin==0:
1241. self.StateSound()
1242. elif self.userChoiceCards == 1:
1243. self.aboutLogo2 = arcade.Sprite(self.detectivePath4 + "1.png",0.8)
1244. text = "Саша очень любит играть на плошадке перед домом,"
1245. text2 = "но сегодня мама его не пустила. Найди Сашу."
1246. text3 = "Какое у него настроение?"
1247. if self.odin==0:
1248. self.StateSound()
1249. color = self.subtitlecolor
1250. text\_size = 33
1251. x = self.SCREEN\_WIDTH // 7
1252. x2 = self.SCREEN\_WIDTH // 5.5
1253. x3 = self.SCREEN\_WIDTH // 3.5
1254. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 7
1255. y2 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 5.25
1256. y3 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 4
1257. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1258. arcade.draw\_text(text2,x2, y2,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1259. arcade.draw\_text(text3,x3, y3,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1260. self.aboutLogo2.center\_x = self.SCREEN\_WIDTH // 2
1261. self.aboutLogo2.center\_y = y - self.aboutLogo2.height // 1.15
1262. self.aboutLogo2.draw()
1263. files = os.listdir(self.detectivePath5)
1264. self.imgMaskes = arcade.SpriteList()
1265. for i in files:
1266. self.imgMask = arcade.Sprite(self.detectivePath5+i, 1)
1267. self.imgMask.width = 100
1268. self.imgMask.height = 100
1269. self.imgMask.center\_x = 0
1270. self.imgMask.center\_y = 0
1271. self.imgMaskes.append(self.imgMask)
1272. #self.imgAvatars.draw()
1273. # Вывод конкретного спрайта
1274. w=self.imgMaskes.sprite\_list[1].width
1275. h=self.imgMaskes.sprite\_list[1].height
1276. counter = 2
1277. s = 20
1278. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2
1279. y = self.SCREEN\_HEIGHT // 6
1280. for i in range(0,2):
1281. self.imgMaskes.sprite\_list[i].center\_x = x
1282. x += w + s
1283. self.imgMaskes.sprite\_list[i].center\_y = y
1284. counter -=1
1285. if counter <=0:
1286. counter = 1
1287. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2
1288. #y += h + s
1289. self.imgMaskes.sprite\_list[i].draw()
1290. # Определяем попадание курсора на аватар
1291. bottom =self.mouseY > self.imgMaskes.sprite\_list[i].center\_y - self.imgMaskes.sprite\_list[i].height // 2
1292. top = self.mouseY < self.imgMaskes.sprite\_list[i].center\_y + self.imgMaskes.sprite\_list[i].height // 2
1293. left = self.mouseX > self.imgMaskes.sprite\_list[i].center\_x - self.imgMaskes.sprite\_list[i].width // 2
1294. right = self.mouseX < self.imgMaskes.sprite\_list[i].center\_x + self.imgMaskes.sprite\_list[i].width // 2
1295. if (bottom and top) and (left and right):
1296. arcade.draw\_rectangle\_outline(self.imgMaskes.sprite\_list[i].center\_x,self.imgMaskes.sprite\_list[i].center\_y,self.imgMaskes.sprite\_list[i].width,self.imgMaskes.sprite\_list[i].height,color=arcade.color.FELDSPAR)
1297. if self.isMouseDown:
1298. self.otvet=i
1299. if (self.otvet==self.vernotvet) and (self.tri\_one==0):
1300. self.StateSound()
1301. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
1302. self.StateSound()
1303. # Стрелка
1304. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.3 + 35
1305. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1306. self.Strelka.draw()
1307. # Дальше
1308. self.MenuFirst2 = 9
1309. self.MenuLasti = 1
1310. mx = self.mouseX
1311. my = self.mouseY
1312. width = 400
1313. height=15
1314. self.MenuItemSelected = 4
1315. for i in range(0,self.MenuLasti):
1316. text = self.Menu[self.MenuFirst2+i]
1317. text\_size = 35
1318. x = self.SCREEN\_WIDTH // 1.3
1319. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1320. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
1321. color = self.menucolorselected
1322. self.sl=8
1323. self.MenuItemSelected = 97
1324. else:
1325. color = self.menucolor
1326. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
1327. # Рисуем крестик
1328. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
1329. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
1330. self.aboutLogo1.draw()
1331. # Определяем попадание курсора на крестик
1332. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
1333. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
1334. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
1335. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
1336. if (bottom and top) and (left and right):
1337. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.FELDSPAR)
1338. if self.isMouseDown:
1339. self.MenuItemSelected = 98
1340. def drawState8(self):
1341. self.vernotvet=2
1342. # Игра Подбери маску 2
1343. text = "Игра Подбери маску"
1344. self.text = "Подбери маску 2"
1345. color = self.titlecolor
1346. text\_size = 38
1347. x = self.SCREEN\_WIDTH // 2.7
1348. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 12
1349. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1350. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
1351. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
1352. self.imgAvatars.sprite\_list[self.userAvatar].draw()
1353. if self.userChoiceCards==0:
1354. self.aboutLogo3 = arcade.Sprite(self.detectivePath4 + "2\_1.png",0.8)
1355. text = "У Иры взяли игрушку без спроса и играют с ней."
1356. text2 = "Найди где Ира."
1357. text3 = "Как себя чувствует Ира?"
1358. if self.dva==0:
1359. self.StateSound()
1360. elif self.userChoiceCards == 1:
1361. self.aboutLogo3 = arcade.Sprite(self.detectivePath4 + "2.png",0.8)
1362. text = "У Вовы взяли игрушку без спроса и играют с ней."
1363. text2 = "Найди где Вова."
1364. text3 = "Как себя чувствует Вова?"
1365. if self.dva==0:
1366. self.StateSound()
1367. color = self.subtitlecolor
1368. text\_size = 33
1369. x = self.SCREEN\_WIDTH // 7
1370. x2 = self.SCREEN\_WIDTH // 3
1371. x3 = self.SCREEN\_WIDTH // 3.1
1372. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 7
1373. y2 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 5.25
1374. y3 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 4
1375. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1376. arcade.draw\_text(text2,x2, y2,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1377. arcade.draw\_text(text3,x3, y3,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1378. self.aboutLogo3.center\_x = self.SCREEN\_WIDTH // 2
1379. self.aboutLogo3.center\_y = y - self.aboutLogo2.height // 1.15
1380. self.aboutLogo3.draw()
1381. files = os.listdir(self.detectivePath5)
1382. self.imgMaskes = arcade.SpriteList()
1383. for i in files:
1384. self.imgMask = arcade.Sprite(self.detectivePath5+i, 1)
1385. self.imgMask.width = 100
1386. self.imgMask.height = 100
1387. self.imgMask.center\_x = 0
1388. self.imgMask.center\_y = 0
1389. self.imgMaskes.append(self.imgMask)
1390. #self.imgAvatars.draw()
1391. w=self.imgMaskes.sprite\_list[1].width
1392. h=self.imgMaskes.sprite\_list[1].height
1393. counter = 4
1394. s = 20
1395. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2
1396. y = self.SCREEN\_HEIGHT // 6
1397. for i in range(0,4):
1398. self.imgMaskes.sprite\_list[i].center\_x = x
1399. x += w + s
1400. self.imgMaskes.sprite\_list[i].center\_y = y
1401. counter -=1
1402. if counter <=0:
1403. counter = 4
1404. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2
1405. #y += h + s
1406. self.imgMaskes.sprite\_list[i].draw()
1407. # Определяем попадание курсора на аватар
1408. bottom =self.mouseY > self.imgMaskes.sprite\_list[i].center\_y - self.imgMaskes.sprite\_list[i].height // 2
1409. top = self.mouseY < self.imgMaskes.sprite\_list[i].center\_y + self.imgMaskes.sprite\_list[i].height // 2
1410. left = self.mouseX > self.imgMaskes.sprite\_list[i].center\_x - self.imgMaskes.sprite\_list[i].width // 2
1411. right = self.mouseX < self.imgMaskes.sprite\_list[i].center\_x + self.imgMaskes.sprite\_list[i].width // 2
1412. if (bottom and top) and (left and right):
1413. arcade.draw\_rectangle\_outline(self.imgMaskes.sprite\_list[i].center\_x,self.imgMaskes.sprite\_list[i].center\_y,self.imgMaskes.sprite\_list[i].width,self.imgMaskes.sprite\_list[i].height,color=arcade.color.FELDSPAR)
1414. if self.isMouseDown:
1415. self.otvet=i
1416. if (self.otvet==self.vernotvet) and (self.tri\_one==0):
1417. self.StateSound()
1418. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
1419. self.StateSound()
1420. # Стрелка
1421. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.3 + 35
1422. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1423. self.Strelka.draw()
1424. # Дальше
1425. self.MenuFirst2 = 9
1426. self.MenuLasti = 1
1427. mx = self.mouseX
1428. my = self.mouseY
1429. width = 400
1430. height=15
1431. self.MenuItemSelected = 8
1432. for i in range(0,self.MenuLasti):
1433. text = self.Menu[self.MenuFirst2+i]
1434. text\_size = 35
1435. x = self.SCREEN\_WIDTH // 1.3
1436. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1437. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
1438. color = self.menucolorselected
1439. self.sl=9
1440. self.MenuItemSelected = 97
1441. else:
1442. color = self.menucolor
1443. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
1444. # Рисуем крестик
1445. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
1446. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
1447. self.aboutLogo1.draw()
1448. # Определяем попадание курсора на крестик
1449. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
1450. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
1451. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
1452. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
1453. if (bottom and top) and (left and right):
1454. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.FELDSPAR)
1455. if self.isMouseDown:
1456. self.MenuItemSelected = 98
1457. def drawState9(self):
1458. self.vernotvet=5
1459. # Игра Подбери маску 3
1460. text = "Игра Подбери маску"
1461. self.text = "Подбери маску 3"
1462. color = self.titlecolor
1463. text\_size = 38
1464. x = self.SCREEN\_WIDTH // 2.7
1465. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 12
1466. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1467. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
1468. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
1469. self.imgAvatars.sprite\_list[self.userAvatar].draw()
1470. if self.userChoiceCards==0:
1471. self.aboutLogo3 = arcade.Sprite(self.detectivePath4 + "3\_1.png",0.8)
1472. text = "Зоя приехала с родителями в певый раз на море. "
1473. text2 = "Найди где Зоя."
1474. text3 = "Как себя чувствует Зоя?"
1475. if self.tri==0:
1476. self.StateSound()
1477. elif self.userChoiceCards == 1:
1478. self.aboutLogo3 = arcade.Sprite(self.detectivePath4 + "3.png",0.8)
1479. text = "Сёма приехал с родителями в первый раз на море. "
1480. text2 = "Найди где Сёма."
1481. text3 = "Как себя чувствует Сёма?"
1482. if self.tri==0:
1483. self.StateSound()
1484. color = self.subtitlecolor
1485. text\_size = 33
1486. x = self.SCREEN\_WIDTH // 7
1487. x2 = self.SCREEN\_WIDTH // 2.9
1488. x3 = self.SCREEN\_WIDTH // 3.1
1489. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 7
1490. y2 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 5.25
1491. y3 = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 4
1492. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1493. arcade.draw\_text(text2,x2, y2,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1494. arcade.draw\_text(text3,x3, y3,color, text\_size, anchor\_y = "center",font\_name = self.font\_title)
1495. self.aboutLogo3.center\_x = self.SCREEN\_WIDTH // 2
1496. self.aboutLogo3.center\_y = y - self.aboutLogo2.height // 1.15
1497. self.aboutLogo3.draw()
1498. files = os.listdir(self.detectivePath5)
1499. self.imgMaskes = arcade.SpriteList()
1500. for i in files:
1501. self.imgMask = arcade.Sprite(self.detectivePath5+i, 1)
1502. self.imgMask.width = 100
1503. self.imgMask.height = 100
1504. self.imgMask.center\_x = 0
1505. self.imgMask.center\_y = 0
1506. self.imgMaskes.append(self.imgMask)
1507. # Вывод конкретного спрайта
1508. w=self.imgMaskes.sprite\_list[1].width
1509. h=self.imgMaskes.sprite\_list[1].height
1510. counter = 6
1511. s = 20
1512. x = self.SCREEN\_WIDTH // 2 - (counter \* w) //2
1513. y = self.SCREEN\_HEIGHT // 6
1514. for i in range(0,6):
1515. self.imgMaskes.sprite\_list[i].center\_x = x
1516. x += w + s
1517. self.imgMaskes.sprite\_list[i].center\_y = y
1518. counter -=1
1519. if counter <=0:
1520. counter = 6
1521. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 2
1522. #y += h + s
1523. self.imgMaskes.sprite\_list[i].draw()
1524. # Определяем попадание курсора на аватар
1525. bottom =self.mouseY > self.imgMaskes.sprite\_list[i].center\_y - self.imgMaskes.sprite\_list[i].height // 2
1526. top = self.mouseY < self.imgMaskes.sprite\_list[i].center\_y + self.imgMaskes.sprite\_list[i].height // 2
1527. left = self.mouseX > self.imgMaskes.sprite\_list[i].center\_x - self.imgMaskes.sprite\_list[i].width // 2
1528. right = self.mouseX < self.imgMaskes.sprite\_list[i].center\_x + self.imgMaskes.sprite\_list[i].width // 2
1529. if (bottom and top) and (left and right):
1530. arcade.draw\_rectangle\_outline(self.imgMaskes.sprite\_list[i].center\_x,self.imgMaskes.sprite\_list[i].center\_y,self.imgMaskes.sprite\_list[i].width,self.imgMaskes.sprite\_list[i].height,color=arcade.color.FELDSPAR)
1531. if self.isMouseDown:
1532. self.otvet=i
1533. if (self.otvet==self.vernotvet) and (self.tri\_one==0):
1534. self.StateSound()
1535. elif (self.otvet!=self.vernotvet) and (self.tri\_two==0):
1536. self.StateSound()
1537. # Стрелка
1538. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.2 + 35
1539. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1540. self.Strelka.draw()
1541. # Дальше
1542. self.MenuFirst2 = 9
1543. self.MenuLasti = 1
1544. mx = self.mouseX
1545. my = self.mouseY
1546. width = 400
1547. height=15
1548. self.MenuItemSelected = 8
1549. for i in range(0,self.MenuLasti):
1550. text = self.Menu[self.MenuFirst2+i]
1551. text\_size = 35
1552. x = self.SCREEN\_WIDTH // 1.2
1553. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1554. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
1555. color = self.menucolorselected
1556. self.sl=2
1557. self.MenuItemSelected = 97
1558. else:
1559. color = self.menucolor
1560. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
1561. # Рисуем крестик
1562. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
1563. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
1564. self.aboutLogo1.draw()
1565. # Определяем попадание курсора на крестик
1566. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
1567. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
1568. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
1569. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
1570. if (bottom and top) and (left and right):
1571. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.FELDSPAR)
1572. if self.isMouseDown:
1573. self.MenuItemSelected = 98
1574. def drawState5(self):
1575. self.clic = 0
1576. lop=0
1577. # Игра Пятнашки 1
1578. text = "Игра Пятнашки"
1579. color = self.titlecolor
1580. text\_size = 38
1581. x = self.SCREEN\_WIDTH // 2.7
1582. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 12
1583. arcade.draw\_text(text,x, y,color, text\_size, anchor\_y = "center",font\_name = "Comic Sans MS")
1584. self.imgAvatars.sprite\_list[self.userAvatar].center\_x = self.imgAvatars.sprite\_list[self.userAvatar].width
1585. self.imgAvatars.sprite\_list[self.userAvatar].center\_y = self.SCREEN\_HEIGHT - self.imgAvatars.sprite\_list[self.userAvatar].height
1586. self.imgAvatars.sprite\_list[self.userAvatar].draw()
1587. #Создаем сами картинки
1588. files = os.listdir(self.detectivePath1)
1589. self.imgPytns\_1 = arcade.SpriteList()
1590. for j in range(0,4):
1591. for i in files:
1592. k=str(self.RAND\_1[j]+1)+'.jpg'
1593. if str(i)==k:
1594. self.imgPytn = arcade.Sprite(self.detectivePath1+i, 1)
1595. self.imgPytn.width = 150
1596. self.imgPytn.height = 250
1597. self.imgPytn.center\_x = 0
1598. self.imgPytn.center\_y = 0
1599. self.imgPytns\_1.append(self.imgPytn)
1600. #self.imgAvatars.draw()
1601. # Вывод конкретного спрайта
1602. w=self.imgPytns\_1.sprite\_list[1].width
1603. h=self.imgPytns\_1.sprite\_list[1].height
1604. counter = 2
1605. s = 20
1606. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 3
1607. y = self.SCREEN\_HEIGHT // 3
1608. for i in range(0,len(self.imgPytns\_1.sprite\_list)):
1609. self.imgPytns\_1.sprite\_list[i].center\_x = x
1610. x += w + s
1611. self.imgPytns\_1.sprite\_list[i].center\_y = y
1612. counter -=1
1613. if counter <=0:
1614. counter = 2
1615. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 3
1616. y += h + s
1617. #self.imgPytns\_1.sprite\_list[i].draw()
1618. #Создаем "рубашки" карточек
1619. files = os.listdir(self.detectivePath6)
1620. self.imgPytns = arcade.SpriteList()
1621. for i in files:
1622. self.imgPytn = arcade.Sprite(self.detectivePath6+i, 1)
1623. self.imgPytn.width = 150
1624. self.imgPytn.height = 250
1625. self.imgPytn.center\_x = 0
1626. self.imgPytn.center\_y = 0
1627. self.imgPytns.append(self.imgPytn)
1628. #self.imgAvatars.draw()
1629. # Вывод конкретного спрайта
1630. w=self.imgPytns.sprite\_list[1].width
1631. h=self.imgPytns.sprite\_list[1].height
1632. counter = 2
1633. s = 20
1634. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 3
1635. y = self.SCREEN\_HEIGHT // 3
1636. for i in range(0,len(self.imgPytns.sprite\_list)):
1637. self.imgPytns.sprite\_list[i].center\_x = x
1638. x += w + s
1639. self.imgPytns.sprite\_list[i].center\_y = y
1640. counter -=1
1641. if counter <=0:
1642. counter = 2
1643. x = self.SCREEN\_WIDTH // 2 - (counter \* w) // 3
1644. y += h + s
1645. if self.schet\_card == 0:
1646. self.imgPytns.sprite\_list[i].draw()
1647. elif self.schet\_card == 1:''''''
1648. if (self.schet\_card < 2):
1649. for j in range(0,len(self.imgPytns.sprite\_list)):
1650. #print(j)
1651. if self.coin[j] == 1:
1652. lop = k
1653. self.imgPytns\_1.sprite\_list[j].draw()
1654. #arcade.pause(1)
1655. elif self.coin[j] == -1:
1656. pass
1657. else:
1658. self.imgPytns.sprite\_list[j].draw()
1659. elif (self.schet\_card > 2):
1660. self.imgPytns.sprite\_list[i].draw()
1661. self.schet\_card = 0
1662. #arcade.play\_sound(self.music)
1663. self.coin=[0,0,0,0]
1664. elif self.schet\_card == 2:
1665. if k == lop:
1666. #print(k, lop)
1667. #lop="ghj"
1668. self.coin[i]=-1
1669. #print(self.coin)
1670. else:
1671. lop = ""
1672. self.coin=[0,0,0,0]
1673. # Определяем попадание курсора на аватар
1674. bottom =self.mouseY > self.imgPytns.sprite\_list[i].center\_y - self.imgPytns.sprite\_list[i].height // 2
1675. top = self.mouseY < self.imgPytns.sprite\_list[i].center\_y + self.imgPytns.sprite\_list[i].height // 2
1676. left = self.mouseX > self.imgPytns.sprite\_list[i].center\_x - self.imgPytns.sprite\_list[i].width // 2
1677. right = self.mouseX < self.imgPytns.sprite\_list[i].center\_x + self.imgPytns.sprite\_list[i].width // 2
1678. if (bottom and top) and (left and right):
1679. arcade.draw\_rectangle\_outline(self.imgPytns.sprite\_list[i].center\_x,self.imgPytns.sprite\_list[i].center\_y,self.imgPytns.sprite\_list[i].width,self.imgPytns.sprite\_list[i].height,color=arcade.color.FELDSPAR)
1680. if self.isMouseDown:
1681. #print(self.RAND\_1[i]+1)
1682. #self.userAvatar = i
1683. k=str(self.RAND\_1[i]+1)+'.jpg'
1684. #print('номер картрочки ',k)
1685. #print(i)
1686. #print(k)
1687. files1 = os.listdir(self.detectivePath1)
1688. for p in files1:
1689. #print(str(p))
1690. if str(p)==k:
1691. #self.imgPytns.sprite\_list[i]=arcade.Sprite(self.detectivePath6+p, 1)
1692. #self.imgPytns\_1.sprite\_list[i].draw()
1693. self.coin[i]=1
1694. self.schet\_card=self.schet\_card+1
1695. #print("ghj", self.coin)
1696. #print(self.schet\_card)
1697. else:
1698. #self.schet\_card = 0
1699. self.coin[i]=1
1700. '''for c in range(0, len(self.coin)-1):
1701. for s in range(1, len(self.coin)):
1702. #print('ghjgjg', c, 'fggvnb' , s)
1703. if self.RAND\_1[c]==self.RAND\_1[s]:
1704. self.coin[i]=1
1705. self.schet\_card=self.schet\_card+1'''
1706. #print(self.imgPytns.sprite\_list[i])
1707. # Стрелка
1708. self.Strelka.center\_x = self.SCREEN\_WIDTH // 1.3 + 35
1709. self.Strelka.center\_y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1710. self.Strelka.draw()
1711. # Дальше
1712. self.MenuFirst2 = 9
1713. self.MenuLasti = 1
1714. mx = self.mouseX
1715. my = self.mouseY
1716. width = 400
1717. height=15
1718. self.MenuItemSelected = 5
1719. for i in range(0,self.MenuLasti):
1720. text = self.Menu[self.MenuFirst2+i]
1721. text\_size = 35
1722. x = self.SCREEN\_WIDTH // 1.3
1723. y = self.SCREEN\_HEIGHT - self.SCREEN\_HEIGHT // 1.2
1724. if my>y-height and my<y+height and mx>x-width//2 and mx<x+width//2:
1725. color = self.menucolorselected
1726. self.MenuItemSelected = 2
1727. else:
1728. color = self.menucolor
1729. arcade.draw\_text(text, x, y,color, text\_size, font\_name = self.font\_title)
1730. # Рисуем крестик
1731. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH - 0.5\*self.aboutLogo1.width
1732. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT -(0.5\*self.aboutLogo1.height)
1733. self.aboutLogo1.draw()
1734. # Определяем попадание курсора на крестик
1735. bottom =self.mouseY > self.aboutLogo1.center\_y - self.aboutLogo1.height // 2
1736. top = self.mouseY < self.aboutLogo1.center\_y + self.aboutLogo1.height // 2
1737. left = self.mouseX > self.aboutLogo1.center\_x - self.aboutLogo1.width // 2
1738. right = self.mouseX < self.aboutLogo1.center\_x + self.aboutLogo1.width // 2
1739. if (bottom and top) and (left and right):
1740. arcade.draw\_rectangle\_outline(self.aboutLogo1.center\_x,self.aboutLogo1.center\_y,self.aboutLogo1.width,self.aboutLogo1.height,color=arcade.color.FELDSPAR)
1741. if self.isMouseDown:
1742. self.MenuItemSelected = 98
1743. def CountingAnswers(self):
1744. """Обработка верных/неверных отетов"""
1745. if self.otvet==self.vernotvet:
1746. self.userGoodAnswers += 1
1747. #print("верные ответы:", self.userGoodAnswers)
1748. #print("неверные ответы:", self.userBadAnswers)
1749. f = open('vern.txt', 'r')
1750. Alf=f.readlines()
1751. for i in range(0,8):
1752. if self.userAvatar == i:
1753. Alf[i]=str(int(Alf[i])+1)+'\n'
1754. f.close()
1755. #print(Alf)
1756. f = open('vern.txt', 'w')
1757. f.writelines("%s" % j for j in Alf)
1758. f.close()
1759. self.state=self.sl
1760. self.otvet=-1
1761. self.aboutLogo1.center\_x = self.SCREEN\_WIDTH
1762. self.aboutLogo1.center\_y = self.SCREEN\_HEIGHT
1763. self.aboutLogo1.draw()
1764. else:
1765. self.userBadAnswers += 1
1766. #print("верные ответы:", self.userGoodAnswers)
1767. #print("неверные ответы:", self.userBadAnswers)
1768. f = open('nevern.txt', 'r')
1769. Alf=f.readlines()
1770. for i in range(0,8):
1771. if self.userAvatar == i:
1772. Alf[i]=str(int(Alf[i])+1)+'\n'
1773. f.close()
1774. f = open('nevern.txt', 'w')
1775. f.writelines("%s" % j for j in Alf)
1776. f.close()
1777. self.state=self.sl
1778. self.otvet=-1
1779. def on\_key\_press(self, key, modifiers):
1780. """ Обработка нажатий на кнопки """
1781. if key == arcade.key.F:
1782. # Переключение между полноэкранным режимом и обычным
1783. self.set\_fullscreen(not self.fullscreen)
1784. width, height = self.get\_size()
1785. self.set\_viewport(0, width, 0, height)
1786. if key == arcade.key.S:
1787. # Еще один способ переключеие между полноэкранным режимом и обычным. Разница будет заметна, если разрешение экрана будет меньше чем текущее
1788. self.set\_fullscreen(not self.fullscreen)
1789. self.set\_viewport(0, self.SCREEN\_WIDTH, 0, self.SCREEN\_HEIGHT)
1790. # Обрабатываем клавишу ESCAPE
1791. if key == arcade.key.ESCAPE:
1792. if self.state == 0:
1793. self.close()
1794. quit()
1795. elif self.state > 0 and self.state < 4:
1796. self.state=self.state-1
1797. elif (self.state > 1 and self.state < 6) or (self.state==50) or (self.state == 51):
1798. self.state=2
1799. elif self.state == 6:
1800. self.state=3
1801. elif self.state > 6 and self.state < 10:
1802. self.state=self.state-1
1803. #elif self.state == 9:
1804. # self.state=6
1805. def on\_mouse\_motion(self, x, y, dx, dy):
1806. """ Перемещение мышки """
1807. # Запоминаем текущие координаты мыши и ее смещение
1808. self.mouseX = x
1809. self.mouseY = y
1810. self.mouseDX = dx
1811. self.mouseDY = dy
1812. def on\_mouse\_press(self, x, y, button, modifiers):
1813. """ Когда кнопка мыши нажата """
1814. #print(f"You clicked button number: {button}")
1815. if button == arcade.MOUSE\_BUTTON\_LEFT:
1816. self.bgGUIColor = arcade.color.GREEN
1817. self.isMouseDown = True
1818. #self.count()
1819. def on\_mouse\_release(self, x, y, button, modifiers):
1820. """ Когда кнопка мыши отпущена """
1821. if button == arcade.MOUSE\_BUTTON\_LEFT:
1822. self.tri\_two=0
1823. self.tri\_one=0
1824. #if self.otvet ==
1825. if self.MenuItemSelected == 98:
1826. self.state = self.MenuItemSelected
1827. quit()
1828. if self.MenuItemSelected == 50:
1829. self.state = self.MenuItemSelected
1830. if self.MenuItemSelected == 51:
1831. self.state = self.MenuItemSelected
1832. if self.MenuItemSelected>0 and self.MenuItemSelected <=15:
1833. #print("Перключаемся в состояние %s"%(self.MenuItemSelected))
1834. self.state = self.MenuItemSelected
1835. if self.MenuItemSelected == 97:
1836. self.state = self.MenuItemSelected
1837. self.isMouseDown = False
1838. if self.schet\_card == 1:
1839. self.clic = 1
1840. #self.count()
1841. def main():
1842. """ Main method """
1843. app = TApp(False)
1844. app.setup()
1845. arcade.run()
1846. if \_\_name\_\_ == "\_\_main\_\_":
1847. main()