ТЕХНОЛОГИИ КОМПЬЮТЕРНОГО ТЕСТИРОВАНИЯ

УДК 159.93

ГРАФИЧЕСКИЙ КОНСТРУКТОР ЭКСПЕРИМЕНТАЛЬНЫХ ПРОЦЕДУР ДЛЯ КОМПЬЮТЕРНОГО ТАХИСТОСКОПА

С.Л. Артеменков, С.И. Попков

Представлены концепция, функциональные требования, проект, алгоритмы и описание реализации нового типа программы для графического конструирования экспериментальных процедур, предназначенных для предъявления последовательности кадров зрительных стимулов и регистрации двигательного ответа испытуемых в компьютерном тахистоскопе. Программа обеспечивает простое создание, визуализацию и динамическое изменение средствами HTML неограниченно расширяемой иерархической структуры однородных кадров с системой отношений между ними.

This article presents the concept, functional requirements, design, algorithms and description of a new type of program for the graphic design of experimental procedures meant for the presentation of a sequence of frames of visual stimuli and recording the motor response of the subjects in the computer tachistoscope. The program provides for simple creation, visualization and dynamic change with HTML infinitely expandable hierarchical structure of uniform frames with the system of relations between them.

КЛЮЧЕВЫЕ СЛОВА
Программа, графическое конструирование, экспериментальная процедура, компьютерный тахистоскоп, иерархия, отношения, язык HTML.

1. ВВЕДЕНИЕ

Создание экспериментальных процедур для компьютерного тахистоскопа, предназначенного для проведения психологических экспериментальных исследований с различными процедурами предъявления зрительных стимулов и регистрацией двигательного ответа испытуемого, обычно осуществляется путем прямого задания соответствующих параметров эксперимента либо с применением специальных сред и программ, позволяющих автоматизировать и упростить эту задачу. При этом компьютерный тахистоскоп, как правило, требует наличия специального программного обеспечения для корректной интерпретации результатов, выраженных в форме инструкций и описания экспериментальных процедур, а также для поддержки низкоуровневых операций, зависящих непосредственно от оборудования (например, непосредственная работа с содержимым видеопамяти). Большое разнообразие самих экспериментальных процедур, связанное с частым изменением методики опыта, вида стимулов, их параметров, способа обработки ответных реакций наблюдателя [1], также затрудняет разработку универсальных программ для проведения психологических экспериментов. Поэтому управ-

ление тахистоскопом обычно осуществляется с помощью высокоуровневых программ, содержащих описания загружаемых в память ресурсов (изображения, тексты и т.п.) и определенную последовательность команд, которые определяют процедуру данного эксперимента (последовательность предъявлений стимулов и регистрации ответов). Эти средства, как правило, требуют от экспериментатора умения программировать или применять знания специализированного скриптового языка. В этой связи актуальной является задача освобождения экспериментатора от необходимости знать специфические языки и технологии программирования и вместе с тем обеспечения возможности гибкой и быстрой модификации используемых аппаратно-программных средств в соответствии с меняющимися условиями экспериментов [1].

Целью работы является разработка графического конструктора экспериментальных процедур для компьютерного тахистоскопа, представляющего собой программную систему иерархического графического покадрового монтажа процедур экспериментов с поддержкой иерархической структуры и объединения кадров вместе с системой различных отношений между ними. Такой подход дает экспериментатору возможность достаточно простого и гибкого построения планов и конструирования процедур экспериментов в структурно-наглядной форме без использования жестких процедур выбора параметров шаблонов экспериментов.

2. СОЗДАНИЕ ЭКСПЕРИМЕНТАЛЬНЫХ ПРОЦЕДУР

Организация психофизического эксперимента при исследовании зрительного восприятия обычно связана с планированием процедур предъявления испытуемым разного рода стимулов в той или иной последовательности и регистрации ответов выбранной группы испытуемых. Часто эти процедуры многократно повторяются в разных видах, что позволяет собрать данные для их последующей статистической обработки. Экспериментальная процедура в целом при этом представляет собой так или иначе заранее заданный временной процесс, состоящий из последовательности (ленты) дискретных элементов (кадров) определенной длительности. В силу повторяемости ряды этих элементов могут быть объединены в интегральные образования новых элементов, которые образуют последовательность (ленту кадров) на более высоком уровне иерархии объединения элементов.

Все эти элементы в процедуре, с одной стороны, связаны друг с другом последовательностью своего предъявления. С другой стороны, между элементами разных уровней иерархии могут возникать отношения, которые определяют особенности повторения и чередования различных элементов между собой и в зависимости плана необходимых действий. Таким образом, экспериментальные процедуры определенным образом структурированы, что определяет факторы, которые необходимо учитывать при разработке требований к компьютерной программе разрабатываемого конструктора.

1) Поддержка возможностей создания, гибкого графического представления и редактирования элементов экспериментальной процедуры в виде иерархии лент кадров с учетом повторяемости кадров и наличия отношений между ними.

2) Поддержка структурного подхода к сбору экспериментальных данных. Хотя сам конструктор не работает с выводами, полученными в ходе выполнения экспериментальных процедур, но созданные с его помощью процедуры должны задавать на внутреннем уровне единородную нотацию, структура которой обеспечивает «прозрачность» обрабатываемых данных для упрощения их представления в качестве сопроводительных выходных данных эксперимента.

3) Переносимость и адаптивность - независимость выходных процедур от каких-либо внешних условий проведения эксперимента (корректные условия записи и чтения данных).

4) Поддержка контроля за ходом эксперимента. Должны быть предоставлены средства конструктора, которые имеют как четкую внутреннюю логику, так и корректное, с точки зрения экспериментатора, внешнее представление (задание формата и содержимого выходной и входной информации по отношению к испытуемому).

Проведенный ранее анализ распространенного программного обеспечения и платформ для создания и проведения экспериментальных процедур показал, что рассмотренные программные аналоги не представляют достаточно полных функциональных возможностей по графическому конструированию иерархических экспериментальных процедур [2]. В этой связи разработка новой программы конструктора экспериментальных процедур является целесообразной.

3. ОБЩИЕ ТРЕБОВАНИЯ К ПРОГРАММЕ КОНСТРУКТОРА

Программа графического конструктора экспериментальных процедур должна удовлетворять ряду требований, сформированных до этапа ее разработки.

1) Полноценный и самодостаточный графический интерфейс – понятный и удобный в использовании интерфейс, графически отражающий все произведенные изменения в структуре проекта эксперимента в реальном времени и не требующий для выполнения основных операций каких-либо специальных расширений, настроек или команд.

2) Иерархическое представление внутренней структуры экспериментальных процедур. Заданная последовательность элементов группируется конструктором в блоки, которым соответствуют элементы следующего уровня иерархии, таким образом, чтобы каждый следующий уровень иерархии содержал в себе предыдущий.

3) Атомарность и полноценность элементов – элементы внутри заданных иерархических последовательностей должны быть единородными, автономными и информативными. Каждый элемент представляет собой структурную единицу, хранящую в себе всю сопроводительную информацию, определяющую характеристики предъявления.

4) Поддержка «вертикальных» и «горизонтальных» связей между элементами. Под «вертикальными» связями подразумевается указание порядка иерархической вложенности и задание дочерней связи для заданного блока по отношению к связанному с ним элементу. Под «горизонтальными» связями понимаются отношения между элементами в рамках одного блока, такие как повторение, чередование и инверсия.

5) Поддержка ранее созданных экспериментальных процедур – любой когда-либо созданный строго в рамках конструктора проект должен быть совместим со средствами редактирования.

6) Поддержка внешнего хранилища изображений и выборки изображений в палитре по типу и заданному в проекте эксперимента разрешению, а также обеспечение единообразия используемого разрешения предъявлений в экспериментах. Построение палитры изображений с поддержкой интеллектуального упорядочивания изображений на основе частоты использования.

7) Функция «защиты от сбоев» – конструктор должен предусматривать автосохранение в случае внесения критических изменений в проект экспериментальной процедуры.

8) Поддержка легко интерпретируемого универсального, структурированного и последовательного выходного формата данных – форма записи проектов экспериментальных процедур обеспечивает максимальную универсальность конструктора относительно различных сред проведения тахистоскопических процедур. Эта универсальность обеспечивается за счет возможности адаптировать разработанную экспериментальную процедуру для целевой модели тахистоскопа [3].

9) Поддержка работы с несколькими процедурами сразу. Интерфейс должен обеспечивать возможность легко переключаться между несколькими открытыми проектами экспериментальных процедур.

10) Поддержка расширенных функций ввода-вывода. Интерфейс должен обеспечивать как операции создания, сохранения, открытия и удаления разработанных экспериментальных процедур, так и возможность работы с кадрами с помощью операций внутреннего буфера обмена.

11) Рандомизация. Интерфейс, в том или ином виде, должен поддерживать функцию непосредственного перемешивания элементов внутри заданной последовательности.

12) Поддержка визуального смещения лент иерархий. Для обеспечения наглядности и гибкости в процессе редактирования необходимо наделить конструктор функционалом, который позволит перемещать ленты относительно друг друга на требуемое количество кадров.

13) Поддержка сокрытия неактуальных лент иерархий. Каждая лента иерархий, хранящая набор последовательностей, над которыми на данный момент не ведется работа, должна быть наделена возможностью исчезать из обозрения, при этом находясь в памяти на случай, если потребуется ее возвращение для дальнейшей обработки. Механизм сокрытия и возвращения должен быть представлен средствами интерфейса конструктора.

В качестве формы программной реализации конструктора экспериментальных процедур удобно использовать автономное стационарное приложение. В этом виде конструктор может быть достаточно легко интегрирован в комплекс программ конкретной тахистоскопической среды и использован для создания широкого спектра разнообразных экспериментальных планов и процедур.

4. СТРУКТУРА КОНСТРУКТОРА И ПРОГРАММНАЯ ПЛАТФОРМА

Конструктор экспериментальных процедур – достаточно сложная система в смысле наличия большого количества разносторонних функций [4], которые обслуживают процесс построения выходных файлов, хранящих необходимую информацию для построения и выполнения экспериментальных процедур. Тем не менее, эти функции легко можно разделить на два класса слабо связанных друг с другом операций: непосредственно построение экспериментальных процедур и операции ввода-вывода, которые представляют собой набор внутренних команд конструктора, обеспечивающих взаимодействие с операционной системой. Это позволяет сохранять и использовать внесенные в проект процедуры изменения, а также переносить промежуточные данные, описывающие часть какой-либо структуры эксперимента, как внутри проекта, так и между разными проектами.

Анализ ряда трудностей, возникающих при использовании иерархических данных, представлен в [5]. В частности, при решении задач визуализации иерархической структуры, как правило, требуется строить дополнительный слой визуального представления данных, нагружая систему дополнительным, избыточным функционалом, что может привести к серьезным задержкам в работе приложения либо нехватке ресурсов вычислительной системы.

Чтобы избежать или минимизировать эти негативные последствия, визуальное представление структуры данных не должно быть отделено от ее представления на функциональном уровне, чтобы образовывать единую, «прозрачную» систему. Для обеспечения этой функциональности в конструкторе экспериментальных процедур использована двухуровневая структура. Первый уровень представляет собой гипертекстовую платформу для визуализации иерархических систем (с поддержкой функций управления). Второй уровень – это приложение, предоставляющее оболочку для осуществления операций ввода-вывода.

Наиболее современным и часто используемым в силу распространения сети Интернет способом представления структурированных данных является язык HTML – язык гипертекстовой разметки (англ. «HyperText Markup Language»), задаваемый сообщением с типом со-

держимого «text/html» [6]. Не являясь языком программирования, язык гипертекстовой разметки задает лишь представление данных, на основании которого уже браузером, согласно общепринятым веб-стандартам, реализуется конкретный функционал, сопряженный с элементами; этот подход похож на декларативную парадигму программирования в области представления данных. В то же время, заданные с помощью тегов элементы, представляющие структуру данных, образуют иерархически вложенные объекты в модели DOM – динамической модели объектов (англ. «Dynamic Object Model»). Эта модель позволяет рассматривать каждый тег в документе, написанном на языке гипертекстовой разметки, как компонент этой модели, со своими правилами задания атрибутов и событий.

Данные, разрабатываемые с применением языка гипертекстовой разметки, не только иерархически вложены и упорядочены, но и представлены в последовательном формате, который легко конвертировать для любого представления данных. То есть язык гипертекстовой разметки автоматически поддерживает сериализуемость. Кроме того, применение технологии гипертекстовой разметки помогает при работе с данными обеспечивать кроссплатформенность, интегрируемость, удобное визуальное представление и эргономичную реализацию интерфейса, масштабируемость и неограниченность уровней представления структур, атомарность операций и однородность элементов. Поэтому для визуализации иерархических систем в рамках конструктора экспериментальных процедур использована платформа на базе языка разметки гипертекста.

Для корректной работы эта платформа должна удовлетворять определенным требованиям. Прежде всего, не должна вмешиваться в работу внешних по отношению к ней служб и функций приложения. Должна быть обеспечена целостность и независимость реализуемого функционала системы.

С другой стороны, независимость платформы не должна помешать приложению осуществлять «бесшовное», непосредственное взаимодействие с объектами, которые находятся внутри визуализируемой иерархии, а также иметь доступ к их свойствам. Кроме того, операции ввода-вывода, не осуществляемые напрямую (например, запись разработанной экспериментальной процедуры на диск), должны в явном виде поддерживаться в виде некоторого инструментария для внешнего приложения. Из этих условий очевидным образом следует необходимость в разработке достаточно гибкого внешнего интерфейса для осуществления требуемых операций и вызовов функций со стороны стороннего приложения.

Кроме того, необходимость обеспечения единства интерфейсного и функционального слоев создает неизбежные проблемы при интерпретации той части иерархической системы, которую надлежит сохранить или загрузить как экспериментальную процедуру. Необходимо обеспечить отделение функций визуализации от экспериментальных процедур на уровне хранимых данных.

Таким образом, программа конструктора должна обеспечивать работу многофункционального внешнего интерфейса с отделением функций визуализации от экспериментальных процедур на уровне хранимых данных и поддержкой целостности и независимости реализуемого функционала.

5. ПРОЕКТИРОВАНИЕ ПРОГРАММЫ КОНСТРУКТОРА

Проект разработки программы конструктора представлен ниже в нотации UML с использованием диаграммы прецедентов (рис. 1). Программа, состоит из двух основных частей – Приложения, с которым взаимодействует Пользователь, и отдельной Платформы визуализации иерархий. Классы и цикл вызовов на рис. 1 отражают абстрактные группы функций приложения:

1) Класс представления – компонент, задающий инициализацию приложения и цикла вызовов и определяющий все процедуры визуального отображения информации, а также первичный перехват событий, вызванных действиями пользователя.

2) Цикл вызовов – внутренний процесс приложения, который взаимодействует с Пользователем и занимается обработкой событий, перехваченных классом представления либо вызванных промежуточными действиями других компонентов.

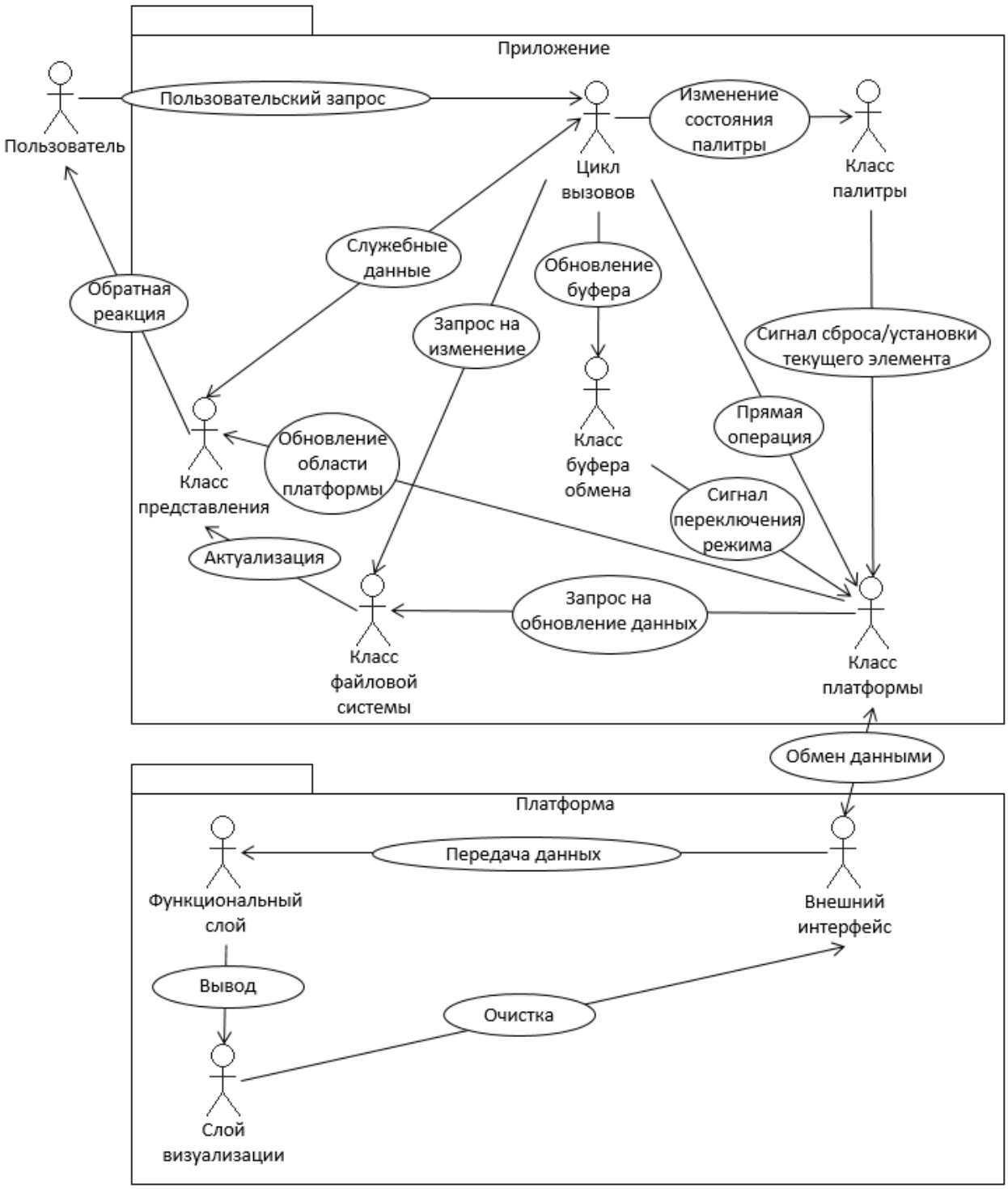

Рис. 1. Общая диаграмма прецедентов.

3) Класс буфера обмена – компонент, который отвечает за работу с данными в области буфера обмена и отправку сигналов, устанавливающих платформу в состояние копирования или вставки, в зависимости от обработанного сигнала в цикле вызовов, а также подготовкой данных из области буфера обмена к перенаправлению на платформу.

4) Класс палитры – компонент, обеспечивающий взаимодействие конечного приложения с хранилищем изображений и данными, задающими положение определенного изображения в палитре, которое, в свою очередь, определяется на основе частоты его использования.

5) Класс файловой системы – один из ключевых компонентов приложения, поскольку отвечает за корректное сохранение либо загрузку файлов проекта экспериментальной процедуры.

6) Класс платформы – компонент, наделяющий приложение функционалом для корректного взаимодействия с внешним интерфейсом платформы.

Платформу (рисунок 1) можно разделить на три части: 1) Функциональный слой – поддерживает работу системы; 2) Слой визуализации – хранит представление иерархии; 3) Внешний интерфейс – промежуточный буфер между системой и приложением.

Пользователь отправляет запрос приложению, которое формирует, в свою очередь, свой внутренний запрос к платформе, дополняя его системными данными. Передача этих данных от внешнего интерфейса к слою визуализации для формирования вывода осуществляется через функциональный слой. При необходимости, по завершении цикла работ внешний интерфейс очищается. Внутреннее взаимодействие между частями платформы отображает диаграмма классов (рис. 2).

Внешний интерфейс содержит в себе поля для обеспечения взаимодействия платформы с вызывающим приложением, а также набор вызовов, гарантирующих корректную работу со свойствами элементов иерархии.

Функциональный слой содержит необходимые для обработки данных функции в качестве операций (работа с цветом для отображения связей, удаление и вставка элементов и строк, модификация таблицы с содержимым и установленных между элементами связей, определение предъявления и прочих данных, принадлежащих заданному в качестве параметра элементу). В качестве атрибутов функционального слоя хранятся промежуточные данные, поддерживающие работу функций со свойствами и связями, а также символ-разделитель для хранимых значений в поле свойств элемента.

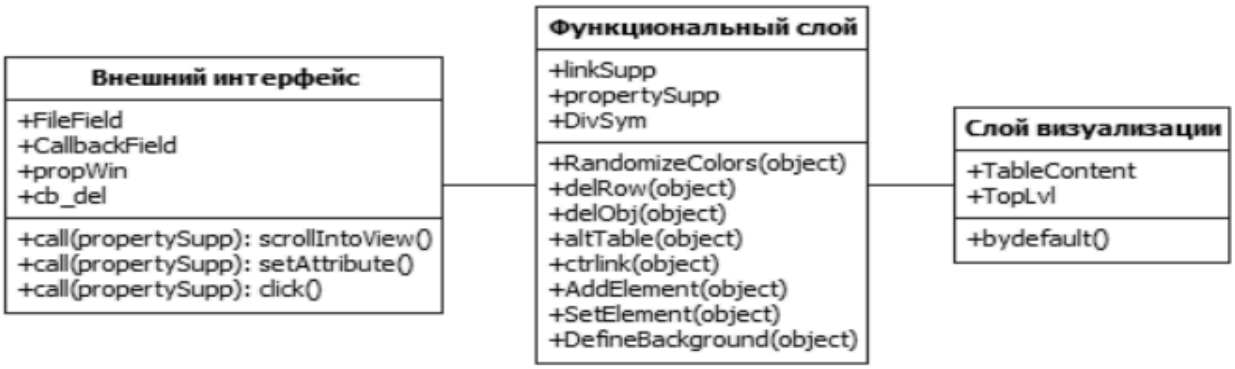

Рис. 2. Диаграмма классов платформы.

Слой визуализации хранит непосредственно содержимое таблицы с данными, а также текущий наивысший уровень иерархии. Из операций здесь существует лишь одна инициализирующая функция, которая задает начальное состояние таблицы до ее модификации. Впоследствии эта функция самоуничтожается, чтобы сохранить целостность данных.

Диаграмма компонентов, приведенная на рис. 3, показывает взаимодействие платформы с приложением и окружающими объектами. В качестве артефактов (или искусственно созданных элементов) предстают как различные файлы, обеспечивающие работу системы, так и хранилище предъявлений.

Субкомпонентом приложения является блок инициализации, которая нужна для подготовки работы приложения с платформой. Как правило, она проводится только один раз. Файл Emul.reg содержит исполняемые инструкции по настройке. После выполнения инициализации приложение передает управление блоку, отвечающему за обработку запросов от пользователя и взаимодействие с платформой. Поддержка самой платформы осуществляется за счет трех файлов: js.js (функциональный слой), def.js (блок определения умолчаний) и h.dat (первоначальный прототип данных о структуре иерархии), которые на выходе преобразуются в кэш текущей веб-страницы с иерархией, находящейся в файле cache.html. Используя этот кэш и внешний буфер обмена данными buf.dat, приложение взаимодействует с функционалом платформы. Хранилище предъявлений непосредственно во взаимодействии не участвует: блок приложения, отвечающий за запросы, самостоятельно выбирает требуемое предъявление для процедуры обработки.

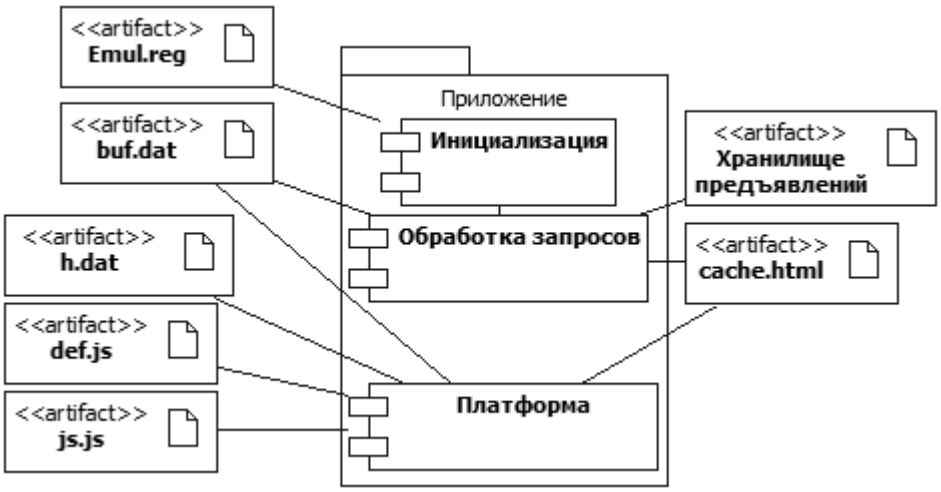

Рис. 3. Диаграмма компонентов платформы и приложения.

Выходные данные представляют собой файл HTML, содержащий основные структурные блоки, выраженные соответствующими тэгами: заголовок, тело документа, системные области для работы со свойствами и элементами, для поддержки поля внешнего интерфейса и уровней иерархии.

Для обеспечения взаимодействия между приложением и интерфейсом разработаны элементы на уровне программного интерфейса: FileField – поле для работы с файлами, хранящими в себе предъявляемое изображение; CallbackField – поле обратного вызова для работы с буфером обмена; propWin – область окна свойств с данными, привязанными к текущему обрабатываемому элементу иерархии; cb_del – флаг переключения в режим модификации/удаления кадра.

Внутренняя форма реализации функций каждого из компонентов приложения показана на диаграмме классов (рис. 4). Представленные на диаграмме атрибуты необходимы для организации взаимодействия между классами.

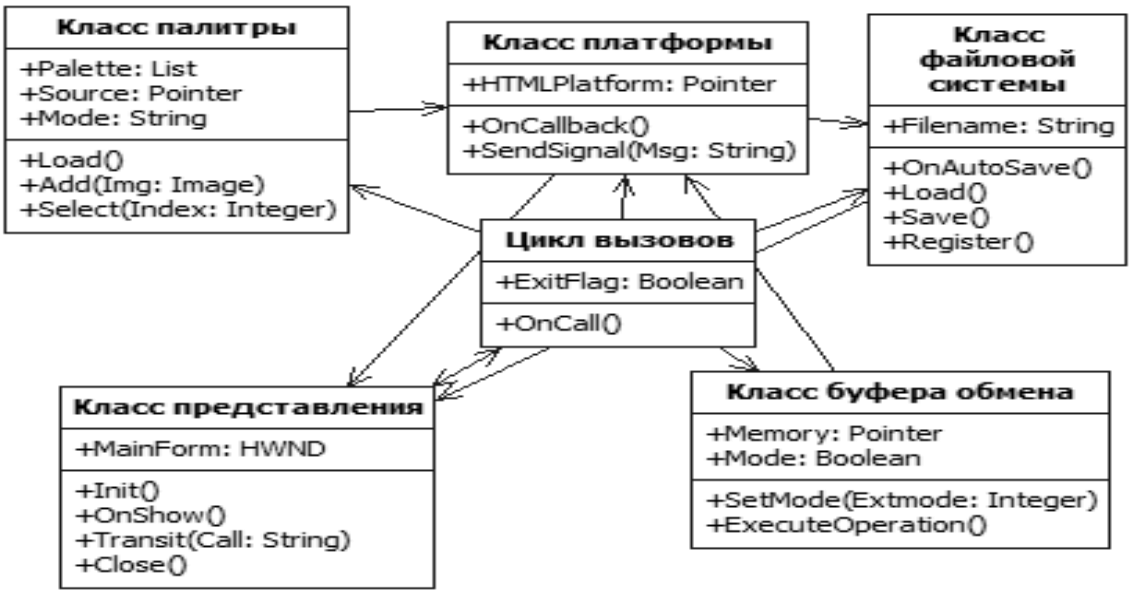

Рис. 4. Диаграмма классов приложения.

1) Класс палитры включает: список объектов палитры Palette, содержащих информацию об изображениях; источник хранилища изображений палитры Source, из которого загружаются изображения и сопутствующая информация; строка Mode, задающая параметры загружаемых изображений; процедура загрузки изображений палитры Load; процедура добавления и регистрации новых изображений в палитре Add; процедура активации изображения палитры Select.

2) Класс платформы включает: указатель на экземпляр платформы HTMLPlatform; обработчик события получения сигнала от слота внешнего интерфейса OnCallback; процедура передачи сигнала внешнему интерфейсу SendSignal.

3) Класс файловой системы включает: имя файла проекта Filename; обработчик события автоматического сохранения проекта OnAutoSave; процедура загрузки проекта Load; процедура записи проекта Save; процедура регистрации проекта в системе Register.

4) Класс буфера обмена включает: указатель на участок временной либо постоянной памяти, которая функционирует как буфер обмена для элементов иерархии проекта, Memory; Mode – логическое (двоичное) значение, определяющее установленный способ работы с буфером (состоянию «истина» соответствует операция вставки, а состоянию «ложь» – копирования); процедура установки текущего режима SetMode; ExecuteOperation – процедура, инициирующая выполнение определенной режимом буфера обмена операции на стороне платформы.

5) Класс представления включает: дескриптор главного окна приложения MainForm; процедура инициализации приложения Init; обработчик события отображения новых сведений средствами интерфейса OnShow; процедура переноса промежуточного события обратно в цикл вызовов Transit; процедура выхода из приложения Close.

6) Цикл вызовов включает: флаг ExitFlag, сигнализирующий о том, что пользователь принял решение завершить работу с приложением; обработчик любого перехваченного события, которое соответствует действию пользователя, OnCall.

Диаграмма компонентов приложения представлена на рис.5.

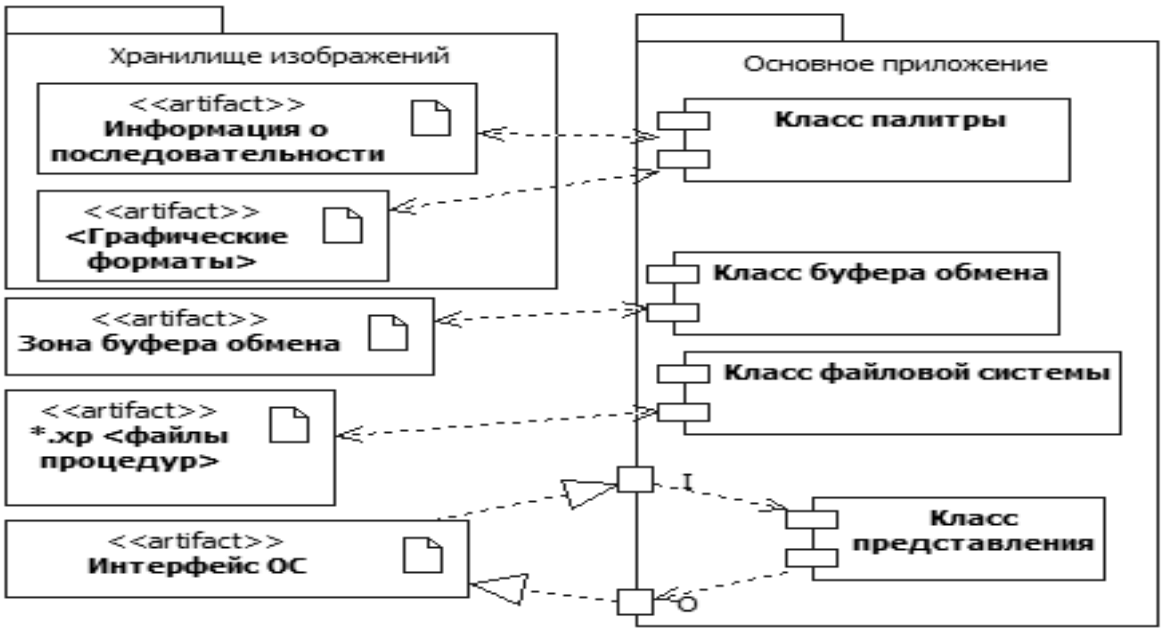

Рис. 5. Диаграмма компонентов приложения.

За процессы ввода-вывода отвечают четыре класса в соответствии с возложенными на них функциями: класс палитры взаимодействует с данными, предоставляемыми подключенным к приложению хранилищем изображений, класс буфера обмена – с буферной областью, класс файловой системы – с файлами проектов экспериментальных процедур, а класс представления – с пользователем и всей совокупностью компонентов приложения.

6. АЛГОРИТМЫ ПРОГРАММЫ КОНСТРУКТОРА

Основной процесс конструктора инициализирует работу программы в целом и поддерживает работу экспериментатора с ней до завершения цикла использования. Общая схема процесса представлена на рис. 6.

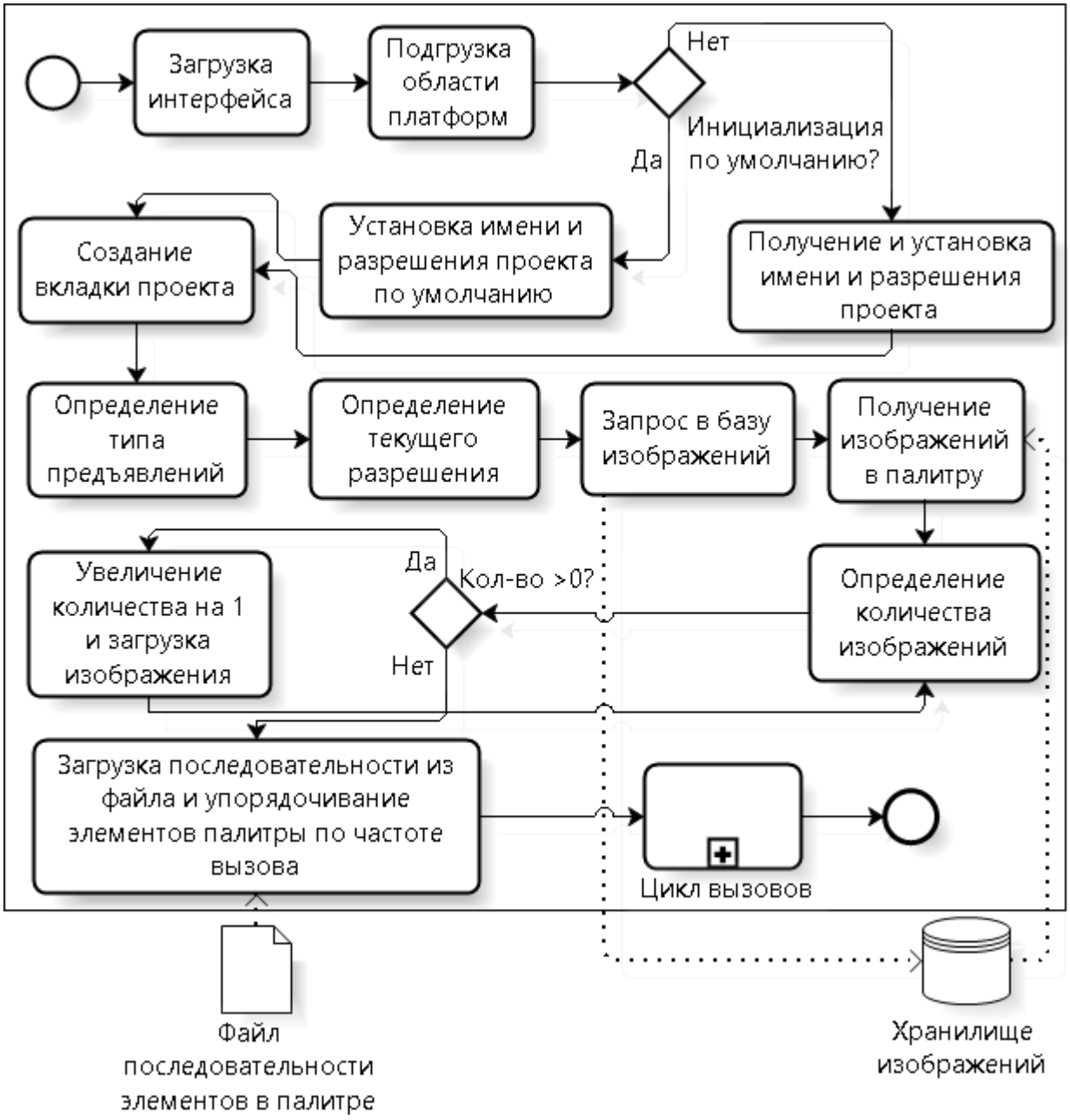

Рис. 6. Основной процесс конструктора.

Процесс работает следующим образом: сначала инициализируется внешний интерфейс конструктора, позволяющий осуществлять операции ввода-вывода, работу с палитрой, а также управлять проектами экспериментальных процедур. Затем происходит инициализация внутренних механизмов, отвечающих за интегрирование платформы в конструктор и представление иерархии экспериментальной процедуры на экране; в свою очередь, на основе этого механизма создается проект с параметрами, установленными по умолчанию. После создания проекта запускается процедура загрузка набора предъявлений. По завершению этой процедуры программа переходит в режим отслеживания событий, реализованный посредством цикла вызовов со стороны пользователя. Когда пользователь осуществляет выход из программы, цикл завершает свое выполнение, после чего программа выгружается из памяти.

За создание проекта отвечает механизм, который осуществляет загрузку умолчаний либо посылает запрос пользователю, после чего создает для нового проекта вкладку, к которой подключается обеспечивающая работу иерархии платформа.

Загрузка набора предъявлений представляет собой расширенный механизм наполнения палитры за счет изображений, находящихся в хранилище, к которому подключен конструктор. Сначала определяется тип предъявляемых изображений и текущее разрешение изображений, поддерживаемое проектом, на основе чего отправляется запрос к хранилищу и осуществляется постепенная выбора требуемых изображений и наполнение палитры. Затем последовательность изображений упорядочивается в соответствии с частотой их использования, которая фиксируется в специальном файле последовательности.

После того, как все необходимые для работы конструктора функции инициализированы, требуется передать управление пользователю. Для этого следует организовать проверку типа и вида события, вызываемого пользовательским вводом (работа с файлами, буфером обмена, палитрой, вкладками и другие операции). За работу с определенными функциями по требованию пользователя отвечает процесс цикла вызовов (рисунок 7).

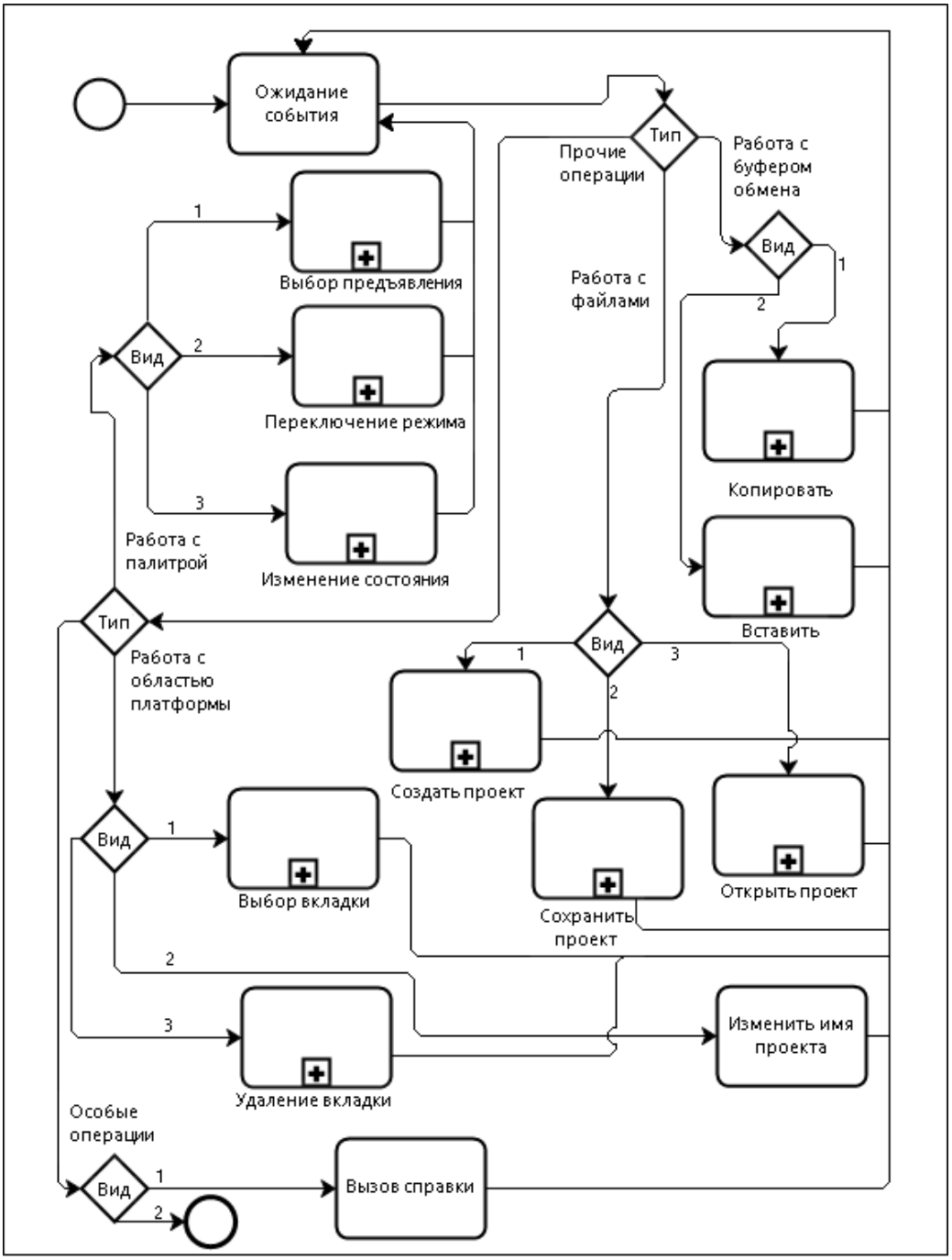

Рис. 7. Процесс цикла вызовов.

7. ОПИСАНИЕ ПРОГРАММЫ КОНСТРУКТОРА

Программа конструктора написана на языке C#, с применением .NET Framework 4.5.1 и технологии тонкого клиента, обеспечивающей интеграцию платформы на базе HTML в качестве редактора иерархических систем.

Для установки программы достаточно распаковать архив с исполняемым файлом, изначальными настройками, основными предъявлениями и набором библиотек для поддержки тонкого клиента. Чтобы распаковать архив, следует запустить файл архива на выполнение, указав целевой путь перед началом процесса извлечения файлов. После этого необходимо

запустить извлеченный исполняемый файл XPD15.exe для загрузки приложения в оперативную память.

Интерфейс основного окна программы конструктора (рисунок 8) состоит из трех областей.

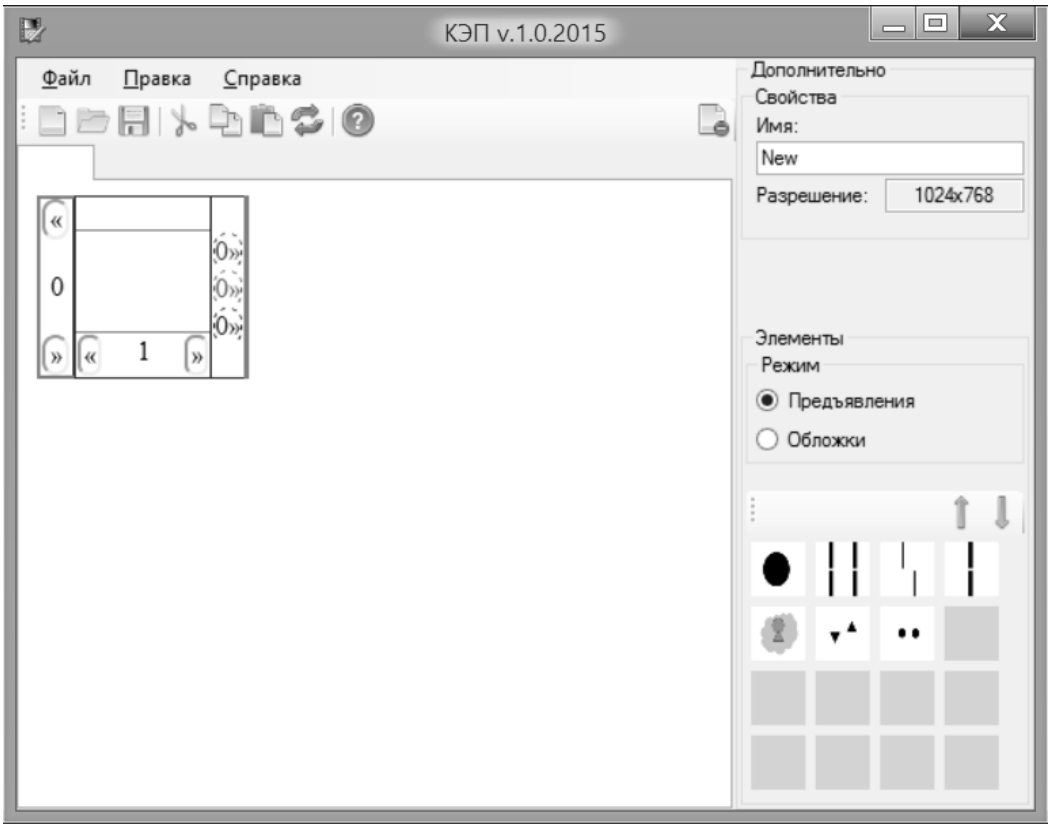

Рис. 8. Основное окно программы.

1) Функциональная область в верхней части окна – состоит из строки меню и панели инструментов. Содержит операции создания либо закрытия вкладки, открытия либо записи проекта, работы с буфером обмена (вырезания, копирования, вставки и вставки с перемешиванием), а также вызова справочной системы.

2) Операционная область – самая большая область окна. Она состоит из одной или нескольких вкладок, а также поля, представляющего собой тонкий клиент для обеспечения работы платформы визуализации иерархий и построения иерархии лент, состоящих из кадров. Вид кадра показан на рисунке 8. Кадр имеет центральную часть, в которой размещается изображение, верхнюю часть для цветового кодирования иерархически вложенной связи, нижнюю часть с функционалом для добавления новых соседних кадров и индексом текущего кадра, а также правую часть с набором связей между кадрами, определяющими порядок предъявления (повторение, чередование, инверсия). Полоса, примыкающая к кадру слева, содержит индекс иерархии и элементы управления для создания и перемещения лент. Представленные элементы управления позволяют строить и изменять экспериментальные процедуры графически средствами визуального интерфейса без использования дополнительных языковых команд.

3) Примыкающая область – серая область с правого края окна. Позволяет определить свойства приложения, работать с палитрой рисунков, выбирать режим работы палитры (предъявления или обложки). Палитра имеет элементы прокрутки (зеленые стрелки) и обеспечивает автоматическое упорядочивание находящихся в ней элементов любого типа по

числу использований. Добавление новых изображений в палитру и их обложек производится путем перетаскивания соответствующих файлов на область палитры.

Интерфейс конструктора позволяет адаптировать его по форме и размерам с помощью обычных процедур изменения окна, определенных в операционной системе Windows. Интерфейс отвечает установленным для него требованиям, включая требованиям дружественности и интуитивности за счет поддержки общепринятых элементов, предоставляемых интерфейсами прикладного программирования операционной системы, и реализации стандартной логики взаимодействия с ними.

8. ЗАКЛЮЧЕНИЕ

Представленный в работе графический конструктор экспериментальных процедур для компьютерного тахистоскопа, представляет собой программную систему иерархического графического покадрового монтажа процедур экспериментов с поддержкой иерархической структуры данных, которая обеспечивает создание зависимостей между различными уровнями представления эксперимента внутри экспериментальных процедур.
Конструктор обеспечивает особую систему графического представления экспериментальных процедур в виде их иерархической структуры, образованной иерархией создаваемых пользователем свободно двигающихся вправо и влево лент, которые состоят из последовательности отдельных кадров процедуры эксперимента с учетом их взаимосвязи и повторения в различном порядке. Эти и ряд других функциональных особенностей конструктора являются уникальными и не имеют аналогов в имеющихся компьютерных системах [1,2].

Разработанная и реализованная программа конструктора представляет собой целостную систему, основанную на платформе визуализации иерархий и удовлетворяет большому числу предъявленным требованиям. Программа предназначена для построения экспериментальных процедур графическим способом без наличия у пользователя навыков работы с какими-либо скриптовыми языками. Отличительными особенностями и преимуществами проекта являются легкость в адаптации выходных форм под различные тахистоскопические системы, а также использование тонкого клиента для поддержки слоя, включающего в себя платформу для визуализации иерархических структур данных как независимого встраиваемого элемента на основе технологии языка разметки гипертекста. Это обеспечивает кроссплатформенную поддержку конструирования экспериментальных процедур не только в рамках разрабатываемого конструктора, но и, при необходимости, в сторонних приложениях.

Результаты работы были представлены на ряде конференций [2-5], а также на выставке «НТТМ-2015».

ЛИТЕРАТУРА

1. Чихман В.Н., Солнушкин С.Д., Пронин С.В., Шелепин Ю.Е., Бондарко В.М. Информационные технологии зрительного эксперимента // Экспериментальная психология в России: традиции и перспективы. - М.: Изд-во «Институт психологии РАН», 2010. – С. 189-194.
2. Артеменков С.Л., Попков С.И. Графическое конструирование иерархических экспериментальных процедур. Естественно-научный подход в современной психологии. – М.: Изд-во «Институт психологии РАН», 2014. – С. 118-125.
3. Артеменков С.Л., Попков С.И. Конструктор экспериментальных процедур для компьютерного тахистоскопа с применением языка разметки гипертекста как средства представления иерархических систем (тезисы). VIII Международный форум информационных технологий «IT FORUM 2020 / Консолидация», 2015. – С. 357-358.
4. Артеменков С.Л., Попков С.И. Конструктор экспериментальных процедур для компьютерного тахистоскопа (тезисы). XII Всероссийская научная конференция «Нейрокомпьютеры и их применение». МГППУ, 2014. – 2 с.
5. Попков С.И., Артеменков С.Л. Программная платформа для графического построения иерархической сети однородных фреймов средствами HTML // XIII Всероссийская научная конференция «Нейрокомпьютеры и их применение». Тезисы докладов. – М: МГППУ, 2015. – С. 83-85.
6. HTML 4.0 Specification [Электронный ресурс]. – URL: www.w3.org/TR/REC-html40-971218 (дата обращения: 01.05.2015 г.).

Работа поступила 25.08.2015