МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ

им. А. И. ГЕРЦЕНА»



09.03.01 – ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ПРОФИЛЬ: “ТЕХНОЛОГИИ РАЗРАБОТКИ   
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ”

**Выпускная квалификационная работа**

Разработка игрового приложения на основе веб-канваса

Обучающегося 4 курса

Очной формы обучения

Абельханова Рината Ринатовича

Научный руководитель:

Кандидат педагогических наук, доцент

Государев Илья Борисович

Санкт-Петербург

2019

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc10872524)

[ГЛАВА I 5](#_Toc10872525)

[АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОДГОТОВКА К СОЗДАНИЮ ПРИЛОЖЕНИЯ 5](#_Toc10872526)

[1.1 Исследование предметной области 5](#_Toc10872527)

[1.2 Обзор первоисточника и классических механик 9](#_Toc10872528)

[ВЫВОДЫ К ГЛАВЕ I 14](#_Toc10872529)

[ГЛАВА II 15](#_Toc10872530)

[ИЗУЧЕНИЕ ТЕХНОЛОГИЙ. ПРОЕКТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ 15](#_Toc10872531)

[2.1 Технологии, используемые в ходе разработки игрового приложения 15](#_Toc10872532)

[2.2 Реализация игрового приложения 32](#_Toc10872533)

[ВЫВОД К ГЛАВЕ II 40](#_Toc10872534)

[ЗАКЛЮЧЕНИЕ 41](#_Toc10872535)

[ЛИТЕРАТУРА 42](#_Toc10872536)

ВВЕДЕНИЕ

С каждым днём всё более заметно, что в современном мире информационные технологии проникли во все сферы деятельности человека. Люди стремительно развивают информационные компьютерные технологии, создавая все более совершенные компьютерные системы, чтобы как облегчить себе жизнь, так и занять свой досуг. Видеоигры, как одно из направлений индустрии развлечений, оказали очень существенное влияние на общество, и в настоящее время игровая индустрия по праву может называться большой и значимой. 18 августа 2008 года в Германии видеоигры официально были признаны видом искусства, а в США видеоигры признали особым видом искусства 5 мая 2011 года [1, c.246-247]. В информационных технологиях можно отметить устойчивую тенденцию для неигрового прикладного программного обеспечения к игрофикации (от англ. gamification). Видеоигры проделали путь в развитии от примитивных аркад до огромных открытых миров. Игры порой несут не только развлекательный характер, но и заставляют задумываться, переживать, поднимать серьезные глобальные или психологические вопросы. Также некоторые представители видеоигровой индустрии могут дарить эмоций не меньше, чем просмотр театральной постановки или киноленты. Над современными видеоиграми работают огромные команды программистов, профессиональные писатели, художники, музыканты, дизайнеры, актёры, постановщики и многие другие. Прибыль от игровых хитов всё чаще превосходит сборы от проката фильмов, как и бюджеты, выделенные на некоторых представителей индустрии видеоигр, уже превысили суммы высокобюджетных фильмов. Но, как и в мире кино, где востребованы не только фильмы с современной дорогостоящей высококлассной компьютерной графикой и спец. эффектами, своё место в игровой индустрии находят небольшие инди‑проекты, созданные небольшими командами независимых разработчиков, а то и вовсе одним человеком. Такие игровые приложения зачастую не обладают передовой графикой или звуковым сопровождением высокого класса, как игры знаменитых компаний, но они часто могут либо нести в себе какие-то инновации в геймплее, либо содержать необычный или цепляющий сюжет, либо могут играть на чувствах ностальгии пользователя, а иногда и всё сразу, и в итоге они также пользуются большим спросом. В мире есть чрезвычайно большое количество любителей видеоигр с разным опытом в этой сфере. Бывают начинающие любители игр, а бывают и ностальгирующие умудренные опытом игроки, поэтому на рынке видеоигр всегда найдётся место проектам самого разного характера. Но далеко не все и не всегда готовы вникать в сложность игровых механик или глубину сюжета, поэтому в последнее время весьма весомую часть рынка занимают простенькие примитивные игровые приложения, которые сразу после запуска интуитивно понятны широкому кругу лиц и не требуют дополнительных пояснений.

Таким образом, **целью работы** будет протестировать возможности веб‑канваса в реализации игровых техник и игрофикации.

Для достижения поставленной цели необходимо решить ряд задач:

* Изучить предметную область индустрии видеоигр.
* Выполнить анализ первоисточника и классических механик.
* Изучить технологии, которые можно задействовать в ходе разработки игрового приложения.
* Спроектировать и реализовать игровое приложение.

ГЛАВА I

АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОДГОТОВКА К СОЗДАНИЮ ПРИЛОЖЕНИЯ

1.1 Исследование предметной области

Разработкой видеоигр называют процесс создания видеоигр (компьютерных игр). Усилия в разработке видеоигр предпринимаются разработчиком, который может быть представлен как фирмой или международной командой, разбросанной по всему миру, так и одним человеком. Зачастую традиционные коммерческие игры разрабатываются командами разработчиков в пределах компании, специализирующейся на играх для консолей (игровых приставок) и персональных компьютеров. Также всё чаще крупные компании обращают своё внимание на мобильный рынок, разрабатывая игры для планшетных компьютеров и смартфонов. Разработка таких игр обходится куда дешевле, но из-за своей доступности эти игры нередко приносят больше прибыли. Как правило, разработка крупномасштабных коммерческих игр для консолей и персональных компьютеров обычно финансируется компанией‑издателем, и для её завершения может потребоваться несколько лет. Издатели по окончании разработки занимаются распространением игры и связанными с этим процессом затратами. Иногда компании-издатели могут содержать внутренние команды разработчиков, или же компания-разработчик может разрабатывать игры за свой счет и распространять их без участия издателей, например, средствами цифровой дистрибуции. Благодаря развитию рынка инди‑игр, многие разработчики видеоигр получили возможность работать над своими игровыми проектами без юридических и финансовых обязательств перед компаниями-издателями. Инди-игры (Indie games, от английского independent video games — «независимые компьютерные игры») обычно занимают меньше времени и денег и могут производиться как отдельными лицами, так и небольшими командами разработчиков. Независимая игровая индустрия находится на подъеме, чему способствует рост новых систем онлайн-дистрибуции, таких как Steam, Epic Games Store, Uplay и многих других, а также рынок мобильных игр для устройств на базе операционных систем Android и iOS [2]. Также за последние несколько лет для разработки видеоигр популярность набрала модель «пожертвований» или краудфандинг (от английского сrowd funding, что можно перевести как народное финансирование) с помощью сервисов, аналогичных Kickstarter. Часто в прошлом известные разработчики игр предлагают создать «идейного наследника» известных игр, если наберётся необходимая на разработку сумма. Вот несколько примеров игр, что наиболее успешно собрали «пожертвования» с помощью такой системы [3]:

* Star Citizen (запрашивали 2 миллиона долларов США, собрали — $27 миллионов).
* Bloodstained: Ritual of the Night (запрашивали 500 тысяч долларов США, собрали — $5.5 миллионов).
* Mighty No. 9 (запрашивали 900 тысяч долларов США, собрали — $4 миллиона).
* Torment: Tides of Numenera (запрашивали 1 миллион долларов США, собрали — более $4.2 миллионов).
* Elite: Dangerous (запрашивали $1.25 миллионов долларов США, собрали — более $1.7 миллионов).

Первые видеоигры, разработанные в 1960-х годах, были некоммерческими. Они требовали работы мейнфреймов (от английского mainframe — большой высокопроизводительный отказоустойчивый компьютер со значительным объёмом как оперативной, так и внешней памяти) и были недоступны для широкой публики. Коммерческая разработка игр началась в 1970-х годах с появлением игровых консолей первого поколения, таких как Magnavox Odyssey, Atari Home Pong, Coleco Telstar, Nintendo Color TV-Game, и ранних домашних компьютеров, таких как Apple I. В то время, благодаря низким ценам и малым возможностям компьютеров, программист в одиночку мог разработать целое законченное игровое приложение. Однако, в конце 80-х и 90-х годов постоянно растущая вычислительная мощность компьютеров и повышенные ожидания геймеров весьма осложнили задачу создания соответствующей актуальным на то время тенденциям игры для консолей или персональных компьютеров в одиночку. В итоге индустрия видеоигр, зародившаяся в середине 1970-х годов как движение энтузиастов, за несколько десятилетий выросла из небольшого рынка в мейнстрим (от английского mainstream — «основное течение») с годовой прибылью в 9.5 миллиардов долларов США в 2007 году и 43.4 миллиардов в 2017 году (согласно ежегодным отчётам Entertainment Software Association) [4]. Во время этого экономически благоприятного периода появилось на свет множество знаменитых компаний-издателей, например, Capcom, Electronic Arts и Activision [5]. Стоимость разработки наиболее крупнобюджетных игр («AAA-видеоигр») уже давно может достигать десятков миллионов долларов США, и в течение последних десятилетий эти цифры непрерывно росли. В 2000 году разработка одного проекта обходилась от одного до четырёх миллионов долларов США. Соответственно росли и сроки разработки, и численность команд разработчиков, задействованных одновременно. Средняя стоимость производства AAA-видеоигр росла. Как правило, средний бюджет таких мультиплатформенных проектов, выпускаемых крупнейшими издателями и продающихся на физических носителях, и зачастую являющихся очередной частью известной серии из некоторого количества игр — составляет от 20 до 50 миллионов долларов США, а бюджет самых выдающихся игр может превышать и 100 миллионов долларов США [6]. Крупнобюджетная игра для двух платформ — PlayStation 3 и Xbox 360 — в 2009 году обходилась примерно в 20 миллионов долларов США, при этом команда разработчиков состояла свыше чем из 100 человек и срок разработки составлял порядка трёх лет. По утверждению Алекса Мура (Alex Moore), геймдизайнера из компании Sumo Digital, если бы цены на игры росли в той же пропорции, в 2012 году игры бы стоили по 1800 долларов США [7]. Иными словами, чтобы окупить возросшие бюджеты при сохранении тех же цен, требовалось продать примерно два миллиона копий одной игры. По исследованию компании Newzoo, чьи исследования заказывают такие гиганты игровой индустрии как Sega, Electronic Arts, Valve, мировая аудитория видеоигр в период с 2015 по 2018 года возросла с 1.2 до 2.3 миллиардов человек. [8]

Основной отличительной чертой игровых приложений от приложений другого рода и других видов досуга является наличие геймплея. Как термин геймплей (от английского gameplay) произошёл в настольных играх и потом стал использоваться и в видеоиграх с развитием оных. Сам термин трудно чётко определить, так как понятие само по себе является комплексным и всеохватывающим. Зачастую геймплеем называют игровой процесс с точки зрения игрока, который включает в себя разные содержательные аспекты видеоигры, например, технические, такие как внутриигровая механика, ряд определённых методов взаимодействия игры с человеком и многие другие. Само понятие геймплея сильно обобщено и часто используется для выражения полученных ощущений в ходе прохождения игры под влиянием совокупности факторов, таких как графика, сюжет и звук [9]. Таким образом, под понятием геймплея можно подразумевать ощущения человека, которые он получает в процессе игры, и направленность действий, которые ему приходится при игре совершать. Геймплей можно охарактеризовать как процесс получения впечатлений и нового опыта игроком. Качество этого процесса зависит от совокупности ряда факторов восприятия, что свойственны игре. Во время игры у человека обычно задействованы визуальное, эмоциональное и слуховое восприятие, поэтому, соответственно, такие факторы как графика, сюжет и звук часто приводят как основные, характеризующие геймплей, факторы. Но эти факторы также свойственны и кинематографу, да и далеко не все игровые приложения могут подразумевать в себе сюжет или могут не иметь звукового сопровождения. Главный фактор, что отличает видеоигры от кино и книг — интерактивность. Именно этот элемент вносит весомое значение в определение геймплея. Таким образом, можно получить следующее понятие геймплея. Геймплей — это признак игрового приложения, который подразумевает под собой квинтэссенцию элементов интерактивности, которые взаимодействует с игроком через графику, звук и сюжет.

Видеоигры классифицируют в основном как по жанрам, так и по количеству возможных игроков в одной игровой сессии. Из-за того, что классификация видеоигр недостаточно систематизирована и критерии принадлежности игры к тому или иному жанру не всегда возможно определить однозначно, в различных источниках информация о жанре определённой игры может различаться. Даже при том, что принадлежность игры почти всегда можно определить к одному из основных жанров, который объединяет в себе множество поджанров, существуют проекты с элементами нескольких жанров, которые могут принадлежать каждому из них. Такие видеоигры причисляют либо к одному из жанров, что является в игре основным, либо сразу ко всем, что присутствуют в игре, если геймплей проекта в равной мере состоит из них [10].

Тем не менее в ходе данной работы было принято решение разрабатывать легко узнаваемое игровое приложение, жанр которого не вызывает вопросов, и название которого широко известно — Тетрис.

1.2 Обзор первоисточника и классических механик

Тетрис — легендарная игра с уникальным сочетанием крайней простоты и аддиктивности. Тетрис, продажи которого трудно точно подсчитать, и, по разным оценкам цифры варьируются от 170 [11] до 495 миллионов проданных копий [12], считается самой продаваемой игрой в истории Видеоигровой индустрии. Первый Тетрис был придуман и написан в 1985 году Алексеем Пажитновым, который являлся в тот момент сотрудником Вычислительного центра при Академии наук СССР, на модном в те времена языке Pascal для компьютера «Электроника-60». Название игры произошло от скрещивания слов «тетрамино» и «теннис». Первая версия игры имела в основе 7 фигур, что стали в итоге классическим набором тетриса. В этой версии фигуры выводились не в виде графических изображений, а с помощью текстовых элементов (рисунок 1), так как у компьютера «Электроника-60» был лишь дисплей, способный отображать только цифры и буквы, без какой-либо графики (рисунок 2). [13]

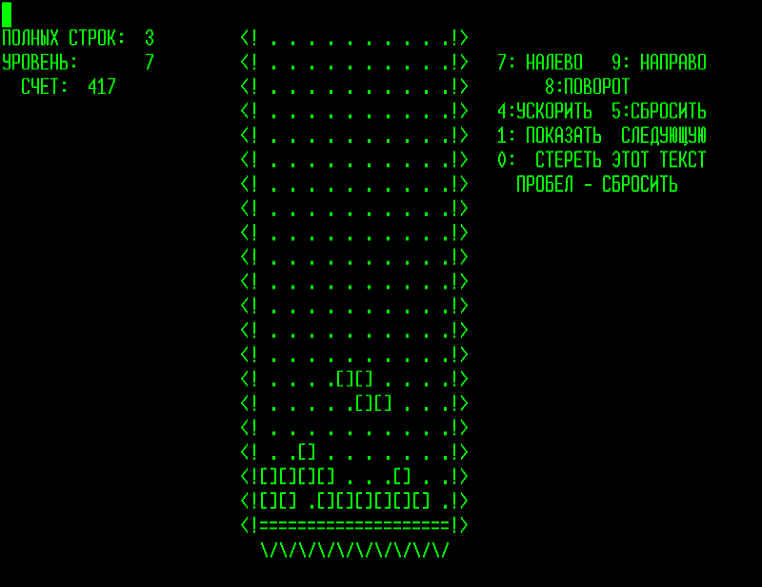


   Рисунок 1 — первый Тетрис    Рисунок 2 — Электроника-60

Во второй версии игры, что была разработана для персональных компьютеров, фигуры уже были цветные и появилась таблица рекордов (рисунок 3).



Рисунок 3 — второй Тетрис

Со временем появилось около 200 официальных версий тетриса [14], каждая из которых могла внести что-то новое, а подсчитать все версии для калькуляторов и прочих электронных устройств, производимых в Китае без какой-либо лицензии, не представляется возможным.

**Геймплей**.

Перед игроком есть поле размером 10 клеток в ширину и 20 клеток в высоту. С верхней части этого поля в случайном порядке падают разные геометрические фигуры, состоящие из 4 одинаковых квадратов, называемые «тетрамино». Пока фигуры падают, игрок может перемещать их по горизонтали и поворачивать на 90 градусов, пока они не коснутся нижней части игрового поля или уже размещенной ранее фигуры. Игрок не может ни предотвратить падение тетрамино, ни замедлить его, но, при желании, может ускорить падение фигуры или моментально сбросить её вниз. Цель игрока состоит в том, чтобы, манипулируя фигурами, образовать сплошную горизонтальную линию без пробелов шириной во всё поле. Как только линия завершена, она исчезает, и блоки, расположенные выше, занимают её место. Как только исчезнет определённое количество линий, игра переходит на новый уровень. С каждым новым уровнем фигуры начинают падать быстрее. Игра заканчивается, как только упавшие фигуры достигают верхней части игрового поля, и следующая фигура не может поместиться. Игра никогда не заканчивается победой игрока. Прежде чем проиграть, игрок должен попытаться набрать как можно больше очков. Чем больше линий исчезнет одновременно, тем больше очков будет начислено игроку. Таким образом, задача игрока — заполнять горизонтальные линии, не заполняя при этом игровое поле, как можно дольше. И чтобы тем самым получить больше очков, нужно складывать фигуры таким образом, чтобы как можно больше линий исчезало одновременно. Как правило, рядом с основным игровым полем рядом находится маленькое поле, в котором отображается какая фигура появится следующей, позволяя игроку тщательней планировать свои действия. Также, в некоторых версиях тетриса, игроку начисляют бонусные очки, если он ускоряет падение тетрамино, тем самым побуждая игрока принимать решения как можно быстрее, и что, соответственно, рано или поздно, приводит к ошибкам.

Тетрис идеально соответствует определению лучших игр: «one minute to learn and a lifetime to master» — Билл Канкел (2003 год) [15] (в переводе с английского: одна минута, чтобы научиться и вся жизнь, чтобы освоить). Игроку требуется максимум минута, чтобы понять, как игровой процесс устроен, но учиться играть в это по-настоящему хорошо можно всю жизнь. В отличие от многих игр, игрок должен строить, а не разрушать. И когда игроку наконец удаётся заполнить линию, она исчезает, оставляя у игрока на виду лишь незаполненные ранее линии и его ошибки. Таким образом, ролью игрока является постоянно исправлять собственные ошибки. [16]

Список существующих тетрамино в стандартной для игр «Тетрис» цветовой схеме изображён на рисунке 4.



Рисунок 4 — список существующих тетрамино

Каждая из фигур тетрамино, за исключением тетрамино I, обязательно занимает два или три ряда в высоту, в зависимости от угла поворота, а значит, каждая из фигур может быть использована для исчезновения как минимум двух полных горизонтальных линий одновременно. Таким же образом, с помощью тетрамино I можно заполнить сразу 4 горизонтальных линий. Такая комбинация называется «Тетрис».

ВЫВОДЫ К ГЛАВЕ I

1. Проведен анализ предметной области игровой индустрии, её возникновения и развития. Определены отличительные черты продуктов данной индустрии.
2. Был изучен первоисточник игрового приложения, его развитие и основные игровые механики.

Во второй главе описаны технологии и процесс создания игрового приложения.

ГЛАВА II

ИЗУЧЕНИЕ ТЕХНОЛОГИЙ. ПРОЕКТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ

2.1 Технологии, используемые в ходе разработки игрового приложения

Было принято решение разрабатывать игровое приложение на основе технологии веб-канваса, так как эта технология позволяет отображать графическое содержимое без лишних подключаемых библиотек практически на всех современных устройствах, если оно имеет поддержку HTML5.

Canvas — это элемент HTML5, используя который можно добавлять растровую графику в вeб-страницу или вeб-приложение с помощью JavaScript. Впервые элемент был использован для создания Mac OS X Dashboard компанией Apple, а затем был реализован в веб-браузерах. Обычно с помощью этого элемента рисуют графики, а также некоторые анимационные составляющие браузерных игр. Также можно иногда наткнуться на мультимедиа элементы, такие как видеоролики, встроенные при помощи Canvas, или даже полноценные браузерные видеоплееры. [17]

HTML5 — последний стабильный релиз, последняя эволюция стандарта, который определяет HTML (от английского HyperText Markup Language — «язык гипертекстовой разметки»). Из простого языка для разметки страницы HTML5 преобразовывает HTML в платформу для создания полноценных приложений. Эта новая версия языка включает в себя новые элементы, атрибуты и поведение, а также расширенный набор технологий, что позволяет создавать более разнообразные и мощные приложения и вeб-сайты. Среди многих особенностей, HTML5 имеет поддержку JavaScript API, которые делают взаимодействия с пользователем, его локальными данными, серверами и предоставляют доступ к аппаратным средствам проще и эффективнее, чем это было раньше. [18]

Улучшению подверглись следующие характеристики:

* **Производительность и интеграция**: более оптимальное использование компьютерного оборудования и предоставление лучшей оптимизации.
  + Новое поколение JavaScript-движков, которые намного мощнее, приводит к приросту производительности.
  + Web Workers позволяет переводить вычисления JavaScript в фоновые потоки, что позволяет предотвращать замедление интерактивных событий.
  + Drag&Drop API (от английского application programming interface — «интерфейс прикладного программирования» или «программный интерфейс приложения») HTML5 позволяет ввести поддержку перетаскивания элементов как внутри веб-сайта, так и между ними.
  + History API предоставляет доступ к истории браузера, что позволяет загружать новую информацию страниц в интерактивном режиме.
  + requestAnimationFrame позволяет управлять рендерингом анимаций для достижения оптимальной производительности.
  + Онлайн и оффлайн события. Может сообщать, когда приложение находится в автономном режиме и когда снова приобретает онлайн статус.
  + Fullscreen API осуществляет контроль использования всего экрана приложениями и веб-страницами без отображения интерфейса браузера.
  + Pointer Lock API позволяет зафиксировать указатель на контент, чтобы игры и прочие интерактивные приложения не теряли фокус, когда указатель достигает края окна.
* **Автономная работа и хранение**: позволяет веб-страницам работать в автономном режиме более эффективно, храня данные локально на стороне клиента. Поддержка IndexedDB, вeб-стандарта для хранения значительных объемов структурированных данных в браузере и для высокопроизводительного поиска по этим данным с использованием индексов. Позволяет пользователю предоставить вeб-приложениям доступ к выбранным локальным файлам.
* **Возможности подключения**: позволяет новыми, инновационными способами обмениваться информацией с серверами.
  + WebRTC (RTC от английского Real-Time Communication — «коммуникация в реальном времени»), технология, которая позволяет подключаться к другим людям и, например, управлять видеоконференцсвязью прямо в браузере без надобности во внешнем модуле или приложении.
  + Веб-сокеты позволяют создать перманентное соединение между сервером и страницей и, с помощью этого, обмениваться не-HTML данными.
  + Позволяет серверу передавать клиенту события, а не следовать классической модели, гд передаются данные только в ответ на запрос клиента.
* **Улучшенная семантика**: позволяет более точно описать, что представляет собой контент.
* **Мультимедия**:Возможность встраивать аудио и видео элементы, использовать и хранить изображения с камеры компьютера, а также задействовать новый мультимедийный контент.
* **2D/3D графика и эффекты**: предоставляет гораздо более широкий набор возможностей отображения информации, с использованием canvas, WebGL и векторных изображений SVG.
* **Доступ к устройству**: позволяет использовать различные устройства ввода и вывода.
  + Обработчики могут реагировать на события, созданные пользователем при нажатии сенсорный экран.
  + Может по согласию пользователя предоставить браузеру доступ к геолокации устройства, чтобы определить местонахождение пользователя.
  + Возможность через браузер обрабатывать изменения положения устройства для адаптации макета страницы или для игр, что реагируют на изменение положения устройства в пространстве.
* **Стиль**: CSS был расширен, чтобы была возможность менять стиль элементов гораздо более сложным способом.
  + Были добавлены две новые компоновки для повышения гибкости дизайна: макеты с гибкой компоновкой и с несколькими столбцами.
  + Типографические улучшение позволяют авторам лучше контролировать переносы и переполнения текста, добавлять тень и более точно настраивать визуальные эффекты и пользовательские шрифты.
  + Управление стилем с помощью анимации. Теперь возможно управлять мобильными элементами на странице используя CSS-анимации для анимирования частей страницы без инициирующего события или с помощью CSS-переходов анимировать переходы между различными состояниями элементов.
  + Теперь можно использовать изображения для стилизации границ и добавлена поддержка их закругления.
  + Также теперь можно наносить тени на элементы, использовать нескольких фонов и CSS-фильтры.

В HTML5, по сравнению с предыдущими версиями, были более точно определены правила синтаксического анализа, которые анализируют значение разметки. До появления HTML5 определялось только значение действительной разметки, и это значило, что, если в разметке была допущена хоть одна небольшая ошибка, а большинство веб-сайтов имеют хотя бы одну, поведение браузеров было трудно предсказуемым, и каждый мог вести себя по-разному. Теперь, если браузеры, совместимые с HTML5, столкнутся с ошибками в разметке, то они должны вести себя одинаково. И хоть это требование очень помогает веб‑разработчикам, так как все современные браузеры теперь используют эти правила синтаксического анализа HTML5, некоторые всё ещё используют браузеры, которые не поддерживают совместимость с HTML5, так что настоятельно рекомендуется соблюдать правильную разметку, так как такой код и легче читать, и поддерживать, и это значительно уменьшает вероятность несовместимости, которая существует в различных старых браузерах. [19]

Основные преимущества разработки игровых приложений на HTML5:

* Игровые приложения, разработанные на HTML5, работают на персональных компьютерах, смартфонах, планшетах, Smart TV и других устройствах, которые имеют поддержку HTML5 совместимых браузеров. Пользователи могут запустить игровое приложение где угодно и когда угодно.
* Возможность в любой удобный момент обновить игровое приложение.
* Возможность задействовать свой собственный метод сбора аналитики.
* Возможность продвигать и рекламировать своё игровое приложение в Интернете также, как и в других средствах массовой информации.
* Возможность более удобно общаться с пользователями.
* Возможность использовать любой удобный сервис обработки платежей.

Технологии, которые с помощью HTML5 можно будет задействовать в дальнейшей разработке игрового приложения на ряду с веб-канвасом [20]:

* **HTML audio** — позволяет легко воспроизводить простые звуковые эффекты и музыку.
* **Web Audio API** — даёт возможность управлять воспроизведением, синтезом и обработкой звука в режиме реального времени.
* **Gamepad API** — позволяет задействовать геймпады (от английского “gamepad”) или другие игровые контроллеры.
* **Touch events** — для обеспечения качественной поддержки сенсорных пользовательских интерфейсов. Предоставляет возможность интерпретировать события взаимодействия с сенсорными устройствами трекпадами и экранами.
* **Full Screen API** — позволяет перевести игровой процесс в полноэкранный режим.
* **Pointer Lock API** — даёт возможность зафиксировать мышь или другое указательное устройство в интерфейсе игрового приложения, дабы избежать случайного переключения на посторонние элементы.
* **SVG** (от английского Scalable Vector Graphics — масштабируемая векторная графика) — возможность создания векторной графики, которая свободно масштабируется вне зависимости от размера или разрешения экрана пользователя.
* **WebGL** — графическая библиотека на основе OpenGL ES 2.0. для создания высокопроизводительной трехмерной графики с аппаратным ускорением.
* **Типизированные массивы** — доступ к необработанным двоичным данным из JavaScript и возможность управлять текстурами графических библиотек, игровыми данными и многим другим.
* **WebRTC** — соединение в реальном времени для управления аудио и видеоданными. Возможность создавать не только чаты и видеоконференции, но и передачу данных между пользователями для многопользовательских игровых сессий.
* **WebSockets** — позволяет подключить игровое приложение к серверу для передачи данных в реальном времени. Идеально подходит для многопользовательских игр, чатов и так далее.
* **File API** и **XMLHttpRequest** — дают возможность получать и отправлять любые необходимые данные с веб-сервера, например, загрузка новых игровых уровней и изображений для предоставления информации о статусе игрового приложения не в реальном времени.
* **IndexedDB** — позволяет хранить пользовательские данные на стороне клиента, будь то компьютер, смартфон или любое другое совместимое устройство.
* **Web Workers** — порождает фоновые потоки, запускающие отдельный JavaScript код для использования многоядерных процессоров.

С элементом canvas довольно просто работать и требуется лишь базовое понимание HTML и JavaScript. Следует помнить о том, что данный элемент может не работать в некоторых старых браузерах, но поддерживается большинством современных браузеров. Стандартный размер элемента 300 пикселей в ширину и 150 пикселей в высоту, но эти характеристики можно легко изменить при помощи HTML-атрибутов height и width. Возможно также выставить размеры произвольно, используя CSS, но во время рендеринга изображение будет менять размер в соответствии с его ориентацией и размером. Для динамического создания графики на веб-канвасе обычно используется javascript context object [21]. Canvas может реагировать на JavaScript события, а также на любые действия пользователя, будь то нажатия кнопок, щелчки мышью или сенсорное управление. Элемент canvas может иметь один или несколько контекстов для рендеринга, создавая и изменяя отображаемое содержимое. Изначально холст прозрачный и пустой. Элемент canvas имеет метод getContext(), который используется для получения контекста визуализации и её функции отображения. getContext() принимает один параметр, тип контекста. Canvas, в отличие от SVG, поддерживает только одну примитивную фигуру — прямоугольник. Другие фигуры могут быть созданы комбинацией одного или большего количества контуров, набором точек, соединенных в линии. Для составления контуров можно использовать функции, которые упрощают составление очень сложных фигур. Ниже представлены три функции рисования прямоугольников в canvas:

* fillRect(x, y, width, height)

Создание заполненного прямоугольника.

* strokeRect(x, y, width, height)

Создание прямоугольного контура.

* clearRect(x, y, width, height)

Очистка прямоугольной области, делая содержимое прозрачным.

Каждая из функций принимает несколько параметров:

* параметры **x**, **y** определяют положение верхнего левого угла прямоугольника относительно начала координат в canvas;
* width(ширина) и height(высота) задают размеры прямоугольника.

Контур (path) —  набор точек, соединённые линиями в отрезки, может образовывать фигуры разной ширины и разного цвета, в том числе изогнутые. Контур может быть закрытым. Создание фигур с помощью контуров происходит в несколько важных шагов:

1. Создаётся контур.
2. Используются команды рисования для отрисовки контура.
3. Закрывается контур.
4. Обработка контура в виде обводки и/или заливки для его отображения.

Вот некоторые функции, которые можно использовать в описанных шагах:

* beginPath()

Создание нового контура. После этой команды следуют команды построения и отрисовки контуров. Каждый вызов этого метода сбрасывает набор, что позволяет начать создавать новые фигуры.

* moveTo()

Перемещает в точку с координатами (x, y) точку нового фрагмента контура.

* lineTo()

Прямой линией соединяет последнюю точку фрагмента контура и точку с координатами (x,y).

* arc(x, y, radius, startAngle, endAngle, anticlockwise)

Добавляет дугу к контуру, с центром в точке (x, y), радиусом r, имеющую угол начала startAngle и конца endAngle, которая рисуется в заданном в аргументе anticlockwise направлении (по умолчанию - по часовой стрелке). В данном методе углы измеряют не в градусах, а в радианах. Для перевода градусов в радианы можно использовать JavaScript-выражение: radians = degrees\*(Math.PI/180).

* bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)

Добавляет к контуру кубическую кривую Безье. Для построения требуется три точки. Первые две точки — контрольные, а третья выступает в качестве конечной. Начальной точкой является последняя точка текущего фрагмента контура, но её можно изменить перед созданием кривой Безье с помощью метода moveTo().

* closePath()

Закрывает контур. Этот метод рисует прямую линию из текущей точки в начальную, чем пытается закрыть фигуру. Если фигура является просто точкой или уже была закрыта, то функция ничего не делает.

* stroke()

Рисует фигуру с обводкой границы.

* fill()

Рисует фигуру с заливкой внутренней области. Автоматически закрывает фигуру.

Чтобы повысить производительность и упростить код, объект Path2D, который доступен в последних версиях браузеров, позволяет кэшировать или записывать эти команды рисования, что позволяет быстро запускать ранее созданные контуры. Конструктор Path2D() возвращает вновь созданный объект Path2D с другим контуром в качестве опционального аргумента (создает копию) или со строкой, состоящей из данных контура SVG. Все методы контуров, такие как moveTo(), rect(), arc() и другие, часть из которых описана выше, также доступны для объектов Path2D.

Path2D API с помощью метода addPath добавляет способ комбинирования контуров. Это может быть полезно для создания фигуры из нескольких контуров.

* Path2D.addPath(path [, transform])

Добавляет к текущему контуру другой контур с опциональной матрицей преобразования.

Еще одна мощная функция нового Canvas Path2D API использует данные SVG для инициализации контура на канвасе. Это может позволить вам передавать данные контура и повторно использовать их как в SVG, так и в холсте. [22]

Для применения цвета к фигуре есть два важных свойства, которые можно использовать: fillStyle и strokeStyle.

* fillStyle = color

Устанавливает стиль для заливки фигур.

* strokeStyle = color

Устанавливает стиль грани фигуры.

В качестве присваемого значения может выступать цвет, объект градиента или паттерна.

По умолчанию цвет заливки и границы — черный (значение CSS цвета #000000). Чтобы значение цвета считалось валидным, оно должно быть представлено строкой и соответствовать CSS <color>. Также можно рисовать прозрачные и полупрозрачные фигуры.  Это делается через установку свойства globalAlpha или присваивания полупрозрачного цвета заливки или границы.

* globalAlpha = transparencyValue

Для применения, указывается значения прозрачности для всех будущих фигур, что будут нарисованы на canvas. Значение полупрозрачности могут быть между 0.0 (полная прозрачность) и 1.0 (полная непрозрачность). По умолчанию установлено значение 1.0. Свойство globalAlpha может быть использовано, если вы хотите рисовать формы с одинаковой прозрачностью, но в иной ситуации, обычно устанавливают прозрачность индивидуально к каждой форме, когда указывают их цвет. Поскольку свойства strokeStyle и fillStyle принимают значения цвета CSS, можно использовать rgba обозначения, чтобы назначить им прозрачный цвет. Функция rgba() аналогична функции rgb(), но содержит один дополнительный параметр, который определяет значение прозрачности этого цвета. Допустимый диапазон также находится между 0,0 (полностью прозрачный) и 1,0 (полностью непрозрачный). Как и в большинстве обычных программ для рисования, фигуры можно заполнять и обводить, используя как линейные, так и радиальные градиенты. Создаётся объект CanvasGradient, с помощью метода addColorStop() назначаются ему цвета, затем этот объект можно присвоить свойствам fillStyle или strokeStyle. [23]

* createLinearGradient(x1, y1, x2, y2)

Создает объект линейного градиента с начальной точкой (x1, y1) и конечной точкой (x2, y2).

* createRadialGradient(x1, y1, r1, x2, y2, r2)

Создает радиальный градиент. Параметры являются две окружности, одна с центром в (x1, y1) и радиусом r1, а другая с центром в (x2, y2) с радиусом r2.

* addColorStop(position, color)

Создает новую цветовую остановку на градиентном объекте. Параметр позиции — это число от 0,0 до 1,0, которое определяет позицию цвета в градиенте. Параметр цвета должен быть строкой, представленной в виде CSS <color>, указывающей цвет, который при этом смещении в переходе градиент должен достичь. Можно добавлять несколько градиентов цвета к градиенту.

При использовании заливки можно предоставить алгоритм правила заполнения, с помощью которого можно определить, находится ли точка внутри или вне пути и, следовательно, заполнена ли она или нет. Это полезно, когда путь пересекает сам себя или является вложенным.

Настройка теней использует только четыре свойства:

* shadowOffsetX = float

Определяет горизонтальное расстояние, на которое тень должна распространяться от объекта. Значением по умолчанию является 0.

* shadowOffsetY = float

Определяет вертикальное расстояние, на которое тень должна распространяться от объекта. Значением по умолчанию является 0.

* shadowBlur = float

Определяет силу эффекта размытия. Значение свойства shadowBlur не соответствует количеству пикселей. Значением по умолчанию является 0.

* shadowColor = color

Значение указывает цвет эффекта тени в виде стандартного обозначения цвета CSS. По умолчанию задан полностью прозрачный черный цвет.

Значения первых трёх свойств не зависят от матрицы преобразования. Свойства shadowOffsetX и shadowOffsetY указывают, как далеко тень должна простираться от объекта. При использовании отрицательных значений, тень будет распространяться вверх или влево. При использовании положительных значений, тень будет распространяться вниз или вправо.

Есть несколько свойств, которые позволяют стилизовать линии:

* lineWidth = value

Устанавливает значение ширины линий, рисуемых в будущем.

* lineCap = type

Определяет внешний вид концов линий.

* + butt (по умолчанию)

Концы линий соответствуют крайним точкам.

* + round

Концы линий округлены.

* + square

Концы линий описаны квадратом с равной шириной и половиной высоты линии.

* lineJoin = type

Определяет внешний вид «углов», где начало и конец линий соединяются.

* miterLimit = value

Устанавливает ограничение на переход, когда две линии соединяются под острым углом, чтобы можно было контролировать толщину перехода.

* setLineDash(segments)

Устанавливает текущей линии пунктир.

* lineDashOffset = value

Указывает, где следует начинать пунктир массива на линии.

Свойство lineJoin определяет, как соединяются два сегмента (линий, дуг или кривых) с ненулевой длиной в форме (сегменты с нулевой длиной, заданные конечные точки и контрольные точки, которые находятся точно в таком же положении — пропускаются).

Для этого свойства есть три возможных значения: round, bevel и miter. По умолчанию установлено значение miter. Настройка lineJoin не действует, если два связанных сегмента имеют одинаковое направление, ведь в этом случае область соединения добавлена не будет.

* round

Радиус заполняемой части скругленных углов равен половине ширины линии, центр этого радиуса совпадает с концами соединённых сегментов.

* bevel

Заполняет дополнительную треугольную область между общей конечной точкой соединённых сегментов и внешними прямоугольными углами каждого сегмента.

* miter

Сегменты соединяются путем расширения их внешних краев для соединения в одной точке с эффектом заполнения дополнительной области в форме ромба.

Метод setLineDash и свойство lineDashOffset устанавливают линиям шаблон штрихов. Метод setLineDash принимает массив чисел, который определяет расстояния для рисования линии и разрыва, а свойство lineDashOffset устанавливает смещение, с которого начинается шаблон.

Одна из самых впечатляющих функций веб-канваса — возможность использования изображений. Они могут быть использованы для динамической композиции фото, в виде фонов для графиков или для спрайтов в играх и так далее. Внешние изображения могут быть использованы в любых форматах, которые поддерживает браузер, например, GIF, PNG или JPEG. Можно использовать даже изображения, созданные другими элементами canvas на той же странице. Импортирование изображений в canvas в основном состоит из двух этапов: задав ссылку на HTMLImageElement объект или для другого canvas элемента как источник. Также можно использовать изображение дав ссылку на URL. Затем для рисования изображения на canvas используется функция drawImage(). [24]

* createPattern(image, type)

Создает и возвращает новый объект шаблона канваса. Параметр image — это CanvasImageSource в роли которого могут выступать другой элемент canvas, HTMLImageElement или элемент <video>. Параметр type — это строка, указывающая, как использовать изображение для создания шаблона, и должен иметь одно из следующих значений:

* repeat

Повторяет изображение в виде плитки в вертикальном и горизонтальном направлениях.

* repeat-x

Повторяет изображение в виде плитки по горизонтали, но не по вертикали.

* repeat-y

Повторяет изображение в виде плитки по вертикали, но не по горизонтали.

* no-repeat

Не повторяет изображение в виде плитки. Используется только один раз.

Важно убедиться, что изображение, которое будет использоваться, загружено перед вызовом этого метода, иначе шаблон может быть нарисован некорректно.

Также есть возможность не только рисовать новые фигуры поверх существующих, но и использовать их для маскировки определенных областей, очистки участков на холсте, не ограничиваясь прямоугольниками метода clearRect().

* globalCompositeOperation = type

Устанавливает тип операции компоновки, применяемой при рисовании новых фигур, где type — это строка, идентифицирующая, какую из двенадцати операций компоновки использовать [25].

Поскольку для управления элементами веб-канваса используется JavaScript, есть возможность создать интерактивные анимации. Самым большим неудобством является то, что, когда фигура уже отрисована, её нельзя двигать. Чтобы изобразить движение необходимо перерисовать фигуру и всё, что было нарисовано до неё. Перерисовка сложных кадров может занять много времени, и скорость сильно зависит от производительности компьютера, на котором она выполняется. Далее перечислены необходимые шаги для того, чтобы нарисовать кадр:

1. Сохранить изначальное состояние канвас.

Если были изменены любые настройки, такие как стили, трансформации и прочие, которые затрагивают состояние канвас и нужно, чтобы это состояние использовалось каждый раз, когда был отрисован кадр, то следует сохранить это состояние.

1. Очистить канвас.

Даже если фигура, которую нужно нарисовать, не занимает всю площадь canvas (как фон, например), то всё, что было нарисовано ранее необходимо стереть. Проще всего это сделать при помощи метода clearRect().

1. Восстановить состояние канвас.

Если состояние было сохранено, его следует восстановить, прежде чем отрисовывать новый кадр

1. Нарисовать анимированные фигуры.

Шаг, на котором собственно отрисовывается кадр.

Фигуры отрисовываются на веб-канвасе либо напрямую — при помощи методов элемента canvas, либо с помощью сторонних функций. Обычно результат становится виден после окончания выполнения скрипта. Но невозможно создать анимацию с циклом for. Это значит, что нужен способ выполнения функций отрисовки через интервалы времени. Управлять такими анимациями можно с помощью функций window.setInterval(), window.setTimeout() и window.requestAnimationFrame(), которые могут быть использованы для вызова некоторой функции, через определённый промежуток времени.

* setInterval(function, delay)

Периодически исполняет указанную функцию каждое значение миллисекунд, указанное в параметре delay.

* setTimeout(function, delay)

Запускает выполнение указанной функции через значение миллисекунд, указанное в параметре delay.

* requestAnimationFrame(callback)

Сообщает браузеру, что нужно выполнить анимацию, и отправляет запрос, чтобы браузер вызвал указанную функцию для обновления анимации перед следующей перерисовкой.

Если никакого взаимодействия с пользователем с элементом не планируется, можно использовать функцию setInterval(), которая многократно выполняет, предоставленный ей код. Если же целью является создать игру, в которой контроль анимации осуществляется клавиатурой, мышью или другим устройством, то необходимо использовать setTimeout(). Используя EventListener, можно перехватывать необходимые действия пользователя и запустить соответствующие функции анимации. Более эффективной для создания анимации является функция requestAnimationFrame, так как новая итерация будет вызываться, когда система готова к отрисовке следующего кадра. Происходит около 60 вызовов в секунду и, если вкладка не активна, количество вызовов уменьшается. [26]

Особенности, которые стоит учитывать перед работой с Canvas можно выделить следующее:

* По умолчанию, стартовая точка находится в левом верхнем углу, а не по центру.
* Следует быть осторожным с размерами холста. Если изменить значения высоты и ширины, элемент создаётся заново, а всё содержимое прежнего элемента исчезает вместе с его настройками.

Преимущества использования веб-канваса:

* Возможность точного пиксельного редактирования за счёт растрового формата.
* Возможность использования фильтров для обработки изображений.
* Наличие аппаратного ускорения.
* Удобство работы с большим количеством элементов.
* Большой ассортимент поддерживаемых библиотек.

2.2 Реализация игрового приложения

HTML с JavaScript с использованием нового элемента в HTML5 — canvas — наиболее привлекательная платформа для простых игр. Он не требует ничего, кроме браузера и должен работать на всех относительно современных платформах.

Приложение разложено на основные блоки: базовый метод помощника, игра с ее очередью событий, рендеринг с механизмом аннулирования и основное приложение. Было спроектировано разделение мест объектов настроек в отдельном файле «settings.js», представлен отдельный конструктор с методами-прототипами, представляющими элементы Tetromino и другие структурные элементы, формально выраженные в виде набора отдельных объектов JavaScript. В первую очередь, это позволяет легко настраивать вещи, которые, обычно, нет возможности настраивать. Например, размер поля игры можно изменять в разумных пределах.

Игровое приложение выглядит симметрично на веб-странице любого размера. Пользователь может настроить размер страницы в любое время, даже во время игры. Раскладка пересчитывается в соответствии с window.innerHeight и размером игрового поля в блоках. Пересчет производится для сохранения размера блока целым числом (не дробным значением), поэтому относительный размер доски по сравнению с внутренней высотой страницы изменяется, чтобы сохранить все значения соотношения сторон за счет переменной игровой области поля. Другими словами, он выполнен в стиле хорошо масштабированного компьютерного приложения. Игровое приложение может быть кастомизировано (настроено) в любой момент. Прежде всего, размер игрового поля в блоке может быть изменен в диапазоне от 5x5 до 50x40. Также можно изменить цвет любой фигуры.

Также была добавлена справка, которую можно отобразить на той же странице в любой момент игрового процесса.

Центральным элементом игры является конструктор Tetromino и два метода объекта-прототипа.

Код начинается с файла «settings.js», включающего в первую очередь HTML, а основной код находится в «application.js».

Объект layout получает основные элементы DOM и реализует оригинальную компоновку и поведение компоновки при изменении размера окна. Следующий объект, game, определяет игровую логику, абстрагированную от графического рендеринга, который делегирован объекту rendering, использующему функцию HTML5 Canvas.

Под всем этим ставится несколько простых базовых функций, за которыми следует основная анонимная функция игры, реализованная в форме IIFE (от английского Immediately Invoked Function Expression — JavaScript функция, которая вызывается сразу же после того, как была определена.) [27], которая помогает сохранить все локальные функции недоступными извне основной функции. Этот шаблон используется во всем коде.

Еще одна проблема, решаемая этим шаблоном проектирования, заключается в разрешении требований строгого режима JavaScript. Это помогает использовать внутренние функции и в то же время помещать основной код в блок try-catch, что важно, особенно для разработки.

Основная функция запускает/инициализирует игру, устанавливает обработчики событий и запускает первый кадр; другие кадры запрашиваются через window.requestAnimationFrame. Использование перехвата исключений ограничено самым верхним уровнем: для каждого обработчика событий и основной функции в соответствии со структурной философией обработки исключений.

**Вращение тетрамино.**

Реализовать основные механики игры Тетриса — довольно несложная задача. Наибольшую сложность вызывает часть, связанная с вращением тетрамино. При попытке выполнить математическое вращение, можно обнаружить, что некоторые тетрамино вращаются вокруг центрального блока (J, L, S, Z и T), в то время как другие вращаются вокруг точки между блоками (O и I). Было принято решение предположить, что все элементы концептуально расположены на сетке 4x4, где каждая ячейка либо занята, либо нет. Если определить занята ли клетка или нет можно с помощью 1 и 0 соответственно, а количество ячеек в такой сетке равняется 16, то можно представить каждый случай в виде простого 16‑битного числа (рисунок 5), чтобы точно определить, как тетрамино должно вращаться.

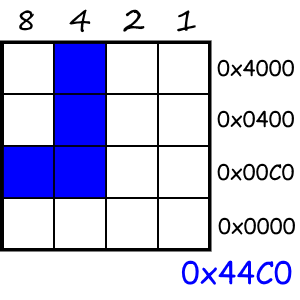


Рисунок 5 — пример 16-битного представления

Oписание бинарного представления каждой формы объекта (рисунок 6):

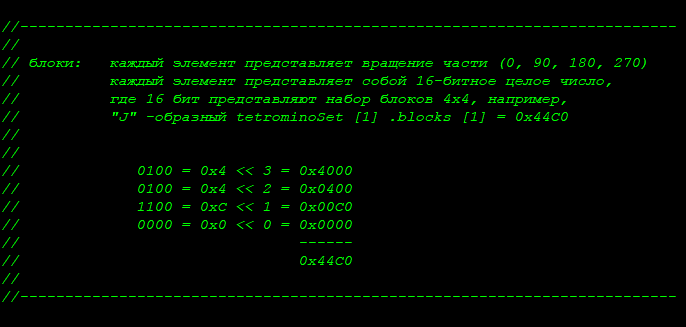


Рисунок 6 — описание бинарного представления

Фрагмент битового определения форм и положения тетрамино:

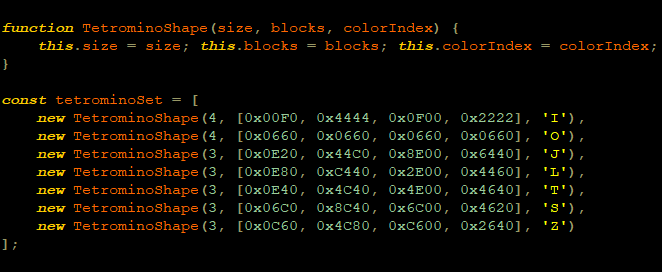


Рисунок 7 — фрагмент определения тетрамино

**Конструктор Тетрамино и Прототипы.**

Был представлен объект конструктора Tetromino по нескольким важным причинам: для повышения производительности кода и, в то же время, сопровождаемости. Связанные функции в JavaScript часто относят к ООП и классам, но это противоречащие термины, что, как минимум, вводит в заблуждение. Механизм объектов на основе прототипов JavaScript принципиально отличается от ООП с классами.

Вот сам конструктор (рисунок 8):

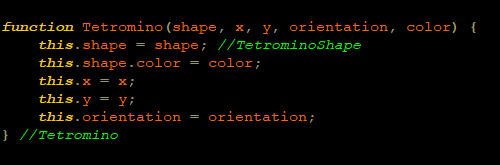


Рисунок 8 — конструктор тетрамино

И вот два метода, что добавлены к его прототипу (рисунок 9):

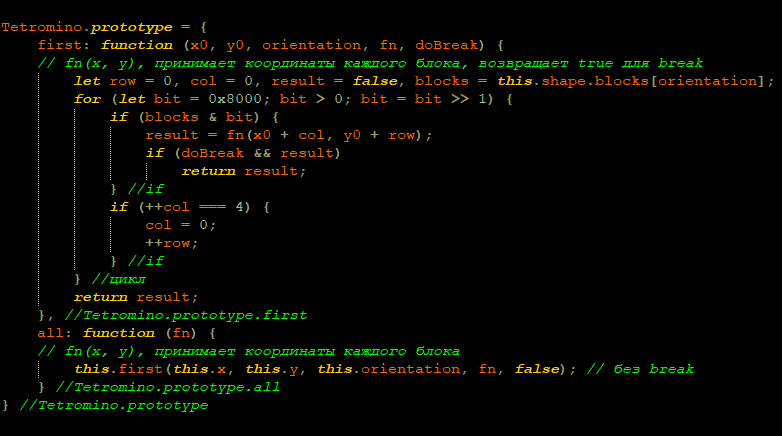


Рисунок 9 — методы прототипа конструктора тетрамино

Эти два метода являются основой низкоуровневых алгоритмов: они реализуют широко известные «first of» и «all» паттерны. Они пересекают все блоки в tetromino shape, и первый из них прерывает поиск, когда какое-либо условие, предоставленное аргументом функции, становится истинным. Это прерывание (break) — одно из основных улучшений производительности, поскольку иначе код проходил бы все блоки заданной формы. Также следует отметить, что first и all создаются только один раз. Именно поэтому это назначение прототипа осуществляется вне конструктора.

Вот как функция first используется в игровой логике (рисунок 10):

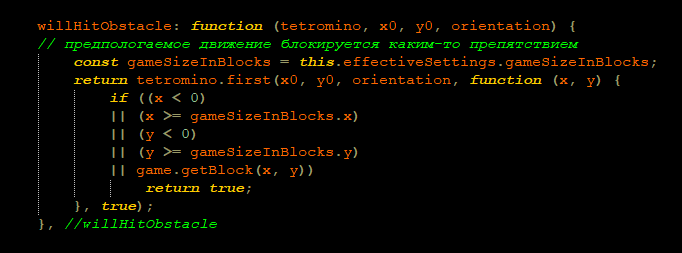


Рисунок 10 — использование функции first

Как только анонимная функция, переданная в tetromino.first, возвращает значение true, функция first также сразу же возвращает значение true, разрывая цикл, проходящий через блоки тетромино. Это указывает на то, что было встречено первое препятствие, которое может быть одной из стен или другим блоком. Обнаружение самого первого препятствия делает излишним дальнейшее рассмотрение препятствий, поэтому функция willHitObstacle возвращает true в этой точке.

Использовать функцию Tetromino.all проще, так как проходятся все блоки фигур. Это используется, в частности, для рисования элементов тетромино на HTML canvas.

**Кастомизация**

Настраиваемая часть игрового приложения находится в отдельном файле "settings.js". Настройки непосредственно из приложения можно вызвать, нажав на клавишу <S>.

1. Размер игрового поля, исчисляемого в блоках, может быть изменён.
2. Управление можно изменить в объекте key. Свойства этого объекта именуются функцией, а не именем ключа. По умолчанию клавиша <Enter> используется для запуска/паузы/продолжения игры, <Esc> прекращает текущую игру, клавиши со стрелками перемещают текущую фигуру, клавиша <вверх> вращает её, кнопка <пробел> сбрасывает фигуру вниз, <F1> показывает и скрывает справку.
3. Скорость игры может быть изменена в объекте delays. Этот объект определяет в секундах задержки до перемещения элемента тетрамино вниз: начальная, минимальная и уменьшение задержки, применяемой для ускорения игры по мере продвижения пользователя. Задержка уменьшается на постоянное значение по мере того, как общее количество линий, согласно правилу игры, увеличивается.
4. Правила начисления очков можно изменить в объекте scoreRules. Эти правила начисляют дополнительные баллы за сброс каждой тетрамино и за удаление строк. Правилами могут быть любые пользовательские функции, рассчитывающие добавленную оценку в зависимости от текущего количества удаленных строк, оценки и количества строк, которые должны быть удалены одновременно. По умолчанию фиксированное количество баллов добавляется для каждого собранного ряда, а количество баллов, добавляемых при удалении линий, увеличивается как степенная функция количества удаленных линий за один раз. Это делается в соответствии с оригинальным игровым дизайном, когда игрок получает стимул собирать количество неполных рядов, а затем заполнять до 4 из них одновременно.
5. Цвета и формы тетрамино могут быть изменены.

**Рендеринг**

Рендеринг становится довольно простым использованием технологии веб‑канваса, с небольшой помощью разметки HTML, так как никаких сложных фигур в игровом приложении нет, всё состоит из квадратов и прямоугольников. Поскольку игровой процесс тетриса движется довольно прерывисто и содержит относительно немного различных деталей, было принято решение, что нет нужды со скоростью 60 кадров в секунду перерисовывать каждый кадр. Достаточно просто перерисовать элементы только тогда, когда они меняются. Для этой простой реализации пользовательский интерфейс был разделён на 4 части и которые перерисовываются только при обнаружении изменений:

* надписи-подсказки
* отображение заработанных очков и исчезнувших рядов
* предварительный просмотр следующего фрагмента
* игровое поле

Последний пункт, игровое поле, является довольно широкой категорией. Технически можно было бы отслеживать каждый отдельный квадрат в сетке и перерисовывать только те, которые изменились, но это было бы излишним. Перерисовка всей сетки может быть выполнена всего за несколько миллисекунд, и если это происходит только тогда, когда произошло изменение, то полная перерисовка сетки происходит лишь 2-3 раза в секунду, на начальном этапе игры, и с продвижением в процессе игры, игровой процесс ускоряется, что соответственно увеличивает количество перерисованных кадров в секунду в приемлемом диапазоне.

ВЫВОД К ГЛАВЕ II

1. Для реализации игрового приложения было проведено изучение технология веб-канваса, в ходе которого стали ясны сильные и слабые стороны технологии.
2. На основе изученной технологии и предшествующего анализа было спроектировано игровое приложение с классической игровой механикой Тетриса.

ЗАКЛЮЧЕНИЕ

В данной работе была рассмотрена предметная область игровой индустрии, изучен первоисточник и развитие Тетриса. Проведен анализ технологий HTML5 и веб-канваса и было спроектировано игровое приложение, отражающее основные игровые механики классического Тетриса на основе технологии веб-канваса и HTML5.

Было достигнуто выполнение следующих задач:

* Проанализирована предметная область игровой индустрии, её возникновение и развитие. Определены характерные черты продуктов данной индустрии.
* Изучен первоисточник игрового приложения, его развитие и основные игровые механики.
* Проведено изучение технологии веб-канваса, в ходе которого определены сильные и слабые стороны технологии.
* Реализовано игровое приложение с классической игровой механикой Тетриса на основе технологии веб-канваса.

Сочетание технологий HTML5 и веб-канваса хорошо подходит для разработки двухмерных мультиплатформенных игровых приложений широкого доступа. Адаптивность технологий под любое современное устройство позволяет избежать большинство проблем портирования подобных проектов.

Использованные технологии позволяют, при необходимости, дополнять реализованное игровое приложение новыми возможностями.

ЛИТЕРАТУРА

1. Богров Ю.С., Видеоигры в контексте глобализации (Эссе) / Наука Телевидения. 2013. №10. С.243-257.
2. Wikipedia, the free encyclopedia — Video game development [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Video_game_development> Дата обращения: 01.04.2019
3. WhatCulture.com [Электронный ресурс]. – Режим доступа: <http://whatculture.com/gaming/10-most-successful-crowdfunded-video-games-of-all-time> Дата обращения: 01.04.2019
4. Entertainment Software Association [Электронный ресурс]. – Режим доступа: [https://www.theesa.com/esa-research/2019-essential-facts-about-the-computer-and-video-game-industry](https://www.theesa.com/esa-research/2019-essential-facts-about-the-computer-and-video-game-industry/) Дата обращения: 21.05.2019
5. Википедия — свободная энциклопедия — Video game industry [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Video_game_industry> Дата обращения: 01.04.2019
6. Kotaku [Электронный ресурс]. – Режим доступа: <https://kotaku.com/how-much-does-it-cost-to-make-a-big-video-game-1501413649> Дата обращения: 01.04.2019
7. Gamasutra [Электронный ресурс]. – Режим доступа: <https://www.gamasutra.com/view/news/168960/Opinion_Unlocking_our_potential.php> Дата обращения: 01.04.2019
8. Newzoo [Электронный ресурс]. – Режим доступа: [https://newzoo.com/insights/articles/newzoos-2018-report-insights-into-the-137-9-billion-global-games-market](https://newzoo.com/insights/articles/newzoos-2018-report-insights-into-the-137-9-billion-global-games-market/) Дата обращения: 21.05.2019
9. Карта Слов [Электронный ресурс]. – Режим доступа: [https://kartaslov.ru/значение-слова/геймплей](https://kartaslov.ru/%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0/%D0%B3%D0%B5%D0%B9%D0%BC%D0%BF%D0%BB%D0%B5%D0%B9) Дата обращения: 30.04.2019
10. Википедия — свободная энциклопедия — Классификация компьютерных игр [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Классификация\_компьютерных\_игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80) Дата обращения: 01.04.2019
11. PC Gamer [Электронный ресурс]. – Режим доступа: [https://www.pcgamer.com/minecraft-has-sold-176-million-copies-may-be-the-best-selling-game-ever](https://www.pcgamer.com/minecraft-has-sold-176-million-copies-may-be-the-best-selling-game-ever/) Дата обращения: 21.05.2019
12. Ferra.ru [Электронный ресурс]. – Режим доступа: <https://www.ferra.ru/review/games/most-succesful-games.htm> Дата обращения: 30.04.2019
13. Игромания [Электронный ресурс]. – Режим доступа: <https://www.igromania.ru/article/11612/Istoriya_tetrisa_Kak_yeto_bylo_na_samom_dele.html> Дата обращения: 30.04.2019
14. Tetris.wiki [Электронный ресурс]. – Режим доступа: <https://tetris.wiki/List_of_official_Tetris_games> Дата обращения: 30.04.2019
15. Gamasutra [Электронный ресурс]. – Режим доступа: <https://www.gamasutra.com/blogs/SebastienHockkoon/20130504/191720/Press_a_Button_to_Fire_Elliptical_Learning_Applied_to_Game_Design.php> Дата обращения: 30.04.2019
16. Wikipédia L’encyclopédie libre — Tetris [Электронный ресурс]. – Режим доступа: <https://fr.wikipedia.org/wiki/Tetris> Дата обращения: 30.04.2019
17. FOXTIME [Электронный ресурс]. – Режим доступа: [https://appfox.ru/blog/kod/canvas-vs-webgl](https://appfox.ru/blog/kod/canvas-vs-webgl/) Дата обращения: 07.05.2019
18. Mozilla Developer Network — HTML5 [Электронный ресурс]. – Режим доступа: [developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5) Дата обращения: 07.05.2019
19. Mozilla Developer Network — Introduction to HTML5 [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/Introduction_to_HTML5> Дата обращения: 07.05.2019
20. Mozilla Developer Network — Introduction to HTML5 Game Development (summary) [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Games/Introduction_to_HTML5_Game_Development_(summary)> Дата обращения: 07.05.2019
21. Mozilla Developer Network — Canvas tutorial [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial> Дата обращения: 07.05.2019
22. Mozilla Developer Network — Drawing shapes with canvas [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shapes> Дата обращения: 07.05.2019
23. Mozilla Developer Network — Applying styles and colors [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Applying_styles_and_colors> Дата обращения: 07.05.2019
24. Mozilla Developer Network — Using images [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Using_images> Дата обращения: 07.05.2019
25. Mozilla Developer Network — Compositing and clipping [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Compositing> Дата обращения: 07.05.2019
26. Mozilla Developer Network — Basic animations [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations> Дата обращения: 07.05.2019
27. Ben Alman [Электронный ресурс]. – Режим доступа: [http://benalman.com/news/2010/11/immediately-invoked-function-expression](http://benalman.com/news/2010/11/immediately-invoked-function-expression/) Дата обращения: 13.05.2019