Шрифт состоит из следующих элементов: вершина, засечка, концевой элемент, петля, вертикальная засечка, каплевидный элемент, точка, верхний выносной элемент, свисание, горизонтальный элемент, узел, нижний выносной элемент, дуга, хвост.

Правильное использование шрифта играет важную роль. Шрифт должен быть не только красивым, но и экономичным. Под гигиеническими правилами к шрифту понимается его удобочитаемость. При выборе основного шрифта нужно принимать во внимание свойства бумаги. Также на выбор шрифта влияет способ печати.

Таким образом, шрифт является одним из важнейших элементов графического дизайна. Шрифтовое оформление играет значимую роль в общем образе издания. Выбор гарнитуры, кегля, начертания непосредственно влияет на читательское восприятие информации.

Список использованных источников

1. Шрифт и шрифтовые композиции [Электронный ресурс] Режим доступа: https://topref.ru/referat/79243.html. – Дата доступа: 28.03.2021

2. Шрифтовые композиции [Электронный ресурс] Режим доступа: https://compuart.ru/article/8880. – Дата доступа: 28.03.2021

3. Шрифт как графическая система [Электронный ресурс] Режим доступа: https://studbooks.net/599184/kulturologiya/shrift_graficheskaya_sistema. – Дата доступа: 28.03.202

УДК 004.921

ПРОБЛЕМА ОПТИМИЗАЦИИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ UNITY

Корзун Д.А., студент
Герасимович О.С., студент
Белорусский национальный технический университет
Минск, Республика Беларусь
Научный руководитель: канд.техн.наук, доцент Дробыш А.А.

Аннотация

В данной научной статье раскрываются способы оптимизации мобильных приложений. В статье рассматривается понятие среды

разработки Unity и её инструментария, а также описываются основные требования к оптимизации мобильных приложений. В заключении статьи подведен общий итог по использованию перечисленных в статье правил и рекомендаций оптимизации приложений.

Мобильные приложения стали неотъемлемой частью жизни человека. Только через Google Play и AppStore доступно более 4 млн различных приложений, и их число с каждым днём растет.

Приложения с каждым годом требуют все больше и больше вычислительных мощностей, что вынуждает разработчиков все чаще задумываться об оптимизации своих проектов.

Мобильные графические процессоры имеют огромные трудности в том, как много тепла они производят, сколько энергии они потребляют, насколько большие или шумные они могут быть. Так же часто бывает, что обработка пикселей в игре ограничивается процессором. Таким образом, в конечном итоге остаются неиспользуемые мощности, особенно на многоядерных процессорах, что негативно сказывается на производительности.

Самым распространенным на данный момент инструментарием для создания приложений под мобильные устройства является Unity

Unity – межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies.

В мире мобильной разработки существуют базовые правила оптимизации, которые необходимо знать каждому Unity разработчику.

Первое правило гласит: не стоит оптимизировать заранее. Существует правило 80/20: 80% - польза, которая получается от 20% проделанной работы. Данные цифры неоднозначны. Чаще всего большая часть оптимизаций, которые производятся в начале разработки проекта, с большой долью вероятности никак не повлияет на конечный проект в целом.

Второе правило гласит: найти то, что нужно оптимизировать. В начале работ над оптимизацией проекта нужно найти то, что тормозит систему, на чем бывают скачки производительности. В этом очень помогает профайлер. Конечно, он довольно условен и сам немного нагружает систему, но польза от него неоспорима.

Третье правило: использовать атласы для комбинирования нескольких текстур в одну большую. Атлас (Atlas) — вид ресурсов, который объединяет несколько текстур в одну. На самом деле важ-

но, чтобы спрайты или модели на сцене использовали один общий материал. Это упрощает доступ к текстурам объектов.

Четвертое правило: не стоит изменять масштаб объекта. Объекты с измененным Scale попадают в отдельную категорию, увеличивающую кол-во Draw Calls. Draw-call — команда на отрисовку от движка к графическому API. Чем больше количество Draw Calls, тем больше нагрузка на видеопроцессор смартфона.

Все вышеперечисленные правила касаются непосредственно настроек самого Unity. Так же важной составляющей хорошей оптимизации является правильное написание кода.

Ниже представлены рекомендации, которые необходимо учитывать в каждом проекте:

— использовать поменьше „ненужных“ циклов. Цикла значительно увеличивают время выполнения скрипта.

— не создавайть переменных в цикле. Лучше создать их за его пределами, а в цикле просто обновлять.

— если есть необходимость обратиться к ресурсам неоднократно, то лучше записать полученный результат в новую переменную и использовать ее. Это позволяет приложению на миллисекунды сократить время обработки запросов.

— по возможности использовать статические методы

— не создавать новые объекты в методах Update(). Метод Update() вызывается на каждый кадр. Чем дольше выполняется данный метод, тем дольше отрисовывается кадр. В идеале он не должен содержать ключевых слов «new»

— создавать объекты один раз и использовать их многократно.

— на финальном этапе программирования необходимо убрать инкапсуляцию (геттеры и сеттеры) и обращаться к переменной напрямую.

Данные правила и рекомендации необходимо учитывать при создании любых мобильных приложений. Так же нужно помнить, что в процессе создания приложения, большую часть времени занимает оптимизация.

Список использованных источников

1. Джосеф Хокинг, Unity в действии / Джосеф Хокинг //Питер. – 2016. – № 2. – С. 118–336.

2. Джереми Гибсон Бонд, Unity в C# геймдев от идеи до реализации / Джереми Гибсон Бонд //Питер. – 2019. – № 2. – С. 329–930.

УДК 004.042

ПАРАЛЛЕЛИЗМ В JAVASCRIPT

Мелихов В.А., студент,
Шнитко А.В., студент
Белорусский национальный технический университет
Минск, Республика Беларусь
Научный руководитель: канд.техн.наук, доцент Дробыш А.А.

Аннотация:
Рассматриваются проблемы использования параллелизма в JavaScript. Продемонстрированы различные способы применения параллелизма.

О выгодах параллельного исполнения процессов на сегодняшний день уже даже не нужно подробно рассказывать, поскольку эта тема многократно освещалась и оказалась достаточно очевидной. На сегодня тенденции таковы, что даже мобильные телефоны оснащаются многоядерными процессорами.

Статистика ресурса Steam демонстрирует явную тенденцию роста числа многопроцессорных систем. К примеру, только 3.1% пользователей владеют однопроцессорной конфигурацией (в 2014 году данное число составляло почти 20%).

Есть и другая тенденция: огромное число приложений переходят в браузеры. Это позволяет обеспечить некоторую универсальность приложения и простоту поддержки, однако, также и неизбежные ограничения.

В связи с этими тенденциями возникает вопрос о возможности эффективного использования ресурсов браузера и JavaScript, в частности – о параллельном исполнении процессов.

Существует исчерпывающее количество способов измерить производительность JavaScript. Эти тесты включают в себя много частей, которые покрывают потенциальные способы использования JavaScript для большого количества вычислений, но, если попытать-