Наука, техника и технологии: глобальные и современные тенденции

I МЕЖДУНАРОДНАЯ НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ

СБОРНИК НАУЧНЫХ ТРУДОВ

НАУЧНАЯ ОБЩЕСТВЕННАЯ ОРГАНИЗАЦИЯ
ПРОФЕССИОНАЛЬНАЯ НАУКА

Наука, техника и технологии: глобальные и современные тенденции

Сборник научных трудов
по материалам I Международной научно-практической конференции

23 мая 2018 г.

www.scipro.ru
Новосибирск, 2018

УДК 001
ББК 72

Главный редактор: Н.А. Краснова
Технический редактор: Ю.О.Канаева

Наука, техника и технологии: глобальные и современные тенденции: сборник научных трудов по материалам I Международной научно-практической конференции, 23 мая 2018 г., Новосибирск: Профессиональная наука, 2018. - 39 с.

ISBN 978-1-387-85809-5

В сборнике научных трудов рассматриваются актуальные вопросы развития техники, инноваций, машиностроения, строительства и т.д. по материалам I Международной научно-практической конференции **«Наука, техника и технологии: глобальные и современные тенденции»**, состоявшейся 23 мая 2018 г. в г. Новосибирск.

Сборник предназначен для научных и педагогических работников, преподавателей, аспирантов, магистрантов и студентов с целью использования в научной работе и учебной деятельности.

Все включенные в сборник статьи прошли научное рецензирование и опубликованы в том виде, в котором они были представлены авторами. За содержание статей ответственность несут авторы.

Электронная версия сборника находится в свободном доступе на сайте www.scipro.ru.

При верстке электронной книги использованы материалы с ресурсов: PSDgraphics

УДК 001
ББК 72

СОДЕРЖАНИЕ

СЕКЦИЯ 1. ИНЖЕНЕРНАЯ ГЕОМЕТРИЯ И КОМПЬЮТЕРНАЯ ГРАФИКА

UDC 004

Abdullayev Z.S., Maxammadjonov M.A., Mahmudjonov A.M. Consciousness of solid modeling

Abdullayev Zafarbek Safibullayevich
Assistant
Namangan State University
Maxammadjonov Maxammadjon Alisher o'g'li
Master's degree student
Tashkent university of information technologies
named by Muhammad Al-Khwarizmi
Mahmudjonov Azizbek Maqsudjon o'g'li
Student
Academic lyceum under
Tashkent university of information technologies
named by Muhammad Al-Khwarizmi

Abstract. *This article decided to solid modeling. Solid modeling is a consistent set of principles for mathematical and computer modeling of three-dimensional solids. Solid modeling is distinguished from related areas of geometric modeling and computer graphics by its emphasis on physical fidelity. Together, the principles of geometric and solid modeling form the foundation of 3D-computer-aided design and in general support the creation, exchange, visualization, animation, interrogation, and annotation of digital models of physical objects.*
Keywords*: modeling, solid modeling, binary tree, 3D modeling.*

The solid model is characterized by a three-dimensional volume occupied by the body it defines. This method of modeling is considered one of the most advanced and reliable when creating copies of real objects.

The advantages of a solid model are:

• Absolute designation of a three-dimensional figure with a probability to separate internal and external areas of an object, which is very important for two-way influence of components;

• Automatic removal of hidden lines;

• Automatically creates 3D sections of components, which is important for a difficult assembly product;

• Analysis methods are used that automatically obtain images of specific weight characteristics by the finite element method;

• You can get tonal effects, and you can manipulate the light source.

There are three methods for creating three-dimensional solid models:

- method of constructive representation (C-Rep);
- boundary presentation method (B-Rep);
- Hybrid modeling.

Method of constructive representation (MCP).

The essence of the method of constructive representation is to create a solid model of standard components, which are called solid-state primitives, and are determined by such parameters as shape, size, anchor point, orientation. The MCP is a binary tree graph G = (V, U), where V is the set of vertices that are the basic elements of the primitive form that are constituents of the object, and U is the set of edges denoting the set-theoretic operations that are performed on the proper base elements form.

The primitives of the model are given by a set of attributes: A = <X, Y, Z, Lx, Ly, Lz, Sx, Sy, ... Sn,> where X, Y, Z - where the coordinates of the anchor point of the local SC to the whole object system; Lx, Ly, Lz are rotation angles, Sx, Sy, ... Sn are the metric parameters of the object. Such Boolean operations are the main toolkit used to determine the relationships of neighboring primitives. These operations are based on the theory of algebraic sets. The most basic operations are union, intersection, subtraction.

Method of boundary representation.

The method of boundary representation is a description of the boundaries of the object being created, or a pointwise analytical task of the faces that describe the object. Perhaps, this is the main method, thanks to which it is possible to design a point, rather than approximate representation of geometric solids. This approach requires the user to specify the boundary and contour of objects, different types of sketches of the object, establishing a tie line between these species, to establish mutual correspondence.

Each of the above methods has its own advantages and disadvantages. The main advantage of the system using the method of constructive representation is the initial formation of the model, since the design of a three-dimensional primitive model of the correct parameters using Boolean operations is a simple task. In addition, the method assumes a brief description of the model in the database. B-rep is usually used to create a complex object that is impossible or very difficult to create using the c-rep method. The C-rep method stores the model as a combination of data and logic algebra procedures, so a small amount of memory is required, but the volume becomes larger due to various calculations that are used to reproduce the model. In models with b-rep representation, the exact boundaries of the model are described, which requires a large amount of memory, but there is no need for computation. Another advantage of the b-rep system is the ease of conversion from the boundary to the wireframe model, and vice versa. This simplicity is due to the fact that the descriptions of borders and wireframe models are similar, which facilitates the process of

transforming models from different forms, and allows systems with b-rep to be combined with other systems.

Since the c-rep and b-rep methods have a huge number of pros and cons, hybrid systems have been created that combine these two methods. Hybrid modeling combines wireframe, surface and solid geometry and uses all possible combinations of parametric and dimensional modeling. Undoubtedly, the use of a single modeling strategy would be the best, but there are situations of using the data previously acquired, or importing data from other systems that have different views. Also, working with wire models or 3D geometry, described surface, is more effective. Sometimes it is better to have different representations for different components. For example, it is better to use surface modeling for a sheet coating, and to simulate a pipeline with an axisymmetric one.

References

1. Beknazarova S.S., Maxammadjonov M.A., Ibodullayev S.N. 3D modeling and the role of 3D modeling in our life, International Scientific and Practical Conference "Innovative Technologies in Science (February 25 – 26, 2016, Dubai, UAE)".
2. Shapiro, Vadim (2001). Solid Modeling.

СЕКЦИЯ 2. РАДИОТЕХНИКА И СВЯЗЬ

УДК 316

Степанчненко Е.А. Проблема масштабируемости сети Биткоин

The problem of network scalability Bitcoin

Степанченко Евгений Андреевич
Студент 2-го курса магистратуры университета СибГУТИ
Научный руководитель
Фионов А.Н. д.т.н., профессор кафедры прикладной математике и кибернетики,
Сибирский государственный университет телекоммуникаций и информатики
Stepanchenko Evgeny Andreevich
Student of the 2nd year of master's degree at the University of SibSUTI

Scientific adviser
Fionov A.N. Doctor of Technical Sciences, Professor of the Department of Applied Mathematics and Cybernetics,
Siberian State University of Telecommunications and Informatics

Аннотация. *Задача технологии Блокчейн, это обмен данными внутри сети, которая гарантирует достоверность информации, не имея при этом центра сети. Текущей проблемой этой сети является скорость обработки данных, масштабируемость.*
Ключевые слова*: Блокчейн, транзакция, сеть, блок, Bitcoin, Ethreum.*
Abstract. *The task of Blockchein technology is the exchange of data within the network, which guarantees the reliability of information, without having a network center at the same time. The current problem of this network is the speed of data processing, scalability.*
Keywords*: Blockchein, transaction, network, block, Bitcoin, Ethreum.*

Технология поучила широкое распространение сначала 2017 года, и активно внедряется во многие информационные сферы деятельности по всему миру. В данной работе я вкратце описываю способ работы этой технологи на реальном примере, уже действующей сети Bitcoin. Также произвожу сравнение сети Bitcoin и Ethereum.

Главная проблема этой технологии, это ее масштабируемость, способность быстро обрабатывать большое количество запросов (транзакций), обеспечивая стабильную работы сети.

Сравнивая показатели сети Bitocoin и Ethereum, я покажу возможные способы увеличение пропускной способности сети основанной на технологии Блокейн.

Технология Блокчейн на примере сети Bitcoin

Биткойн — пиринговая платёжная система, использующая одноимённую единицу для учёта операций и одноимённый протокол передачи данных. Для обеспечения функционирования и защиты системы используются криптографические методы хеширования данных, протокол SHA-256. Вся информация о транзакциях между адресами системы доступна в открытом виде и дублируется каждым участником сети.

Одна из главных особенностей системы — полная децентрализация: нет центрального администратора или какого-либо его аналога. Необходимым и достаточным элементом этой платёжной системы является базовая программа-клиент (имеет открытый исходный код). Запущенные на множестве компьютеров программы-клиенты соединяются между собой в одноранговую сеть, каждый узел которой равноправен и самодостаточен. Невозможно частное управление системой, в том числе изменение суммарного количества биткойнов. Заранее известны объём и время выпуска новых биткойнов, но распределяются они относительно случайно среди тех, кто использует своё оборудование для вычислений, результаты которых являются механизмом регулирования и подтверждения правомочности операций в системе «Биткойн».

Биткойны существуют только в виде записей в реплицированной распределённой базе, в которой в общедоступном открытом (нешифрованном) виде хранятся все транзакции, с указанием биткойн-адресов отправителей/получателей, но без информации о реальном владельце этих адресов. В базе нет отдельных записей о текущем количестве биткойнов у какого-либо владельца. Лишь на основании цепочек транзакций становится понятным текущее количество биткойнов, связанных с тем или иным биткойн-адресом. То есть можно увидеть, что на адрес поступил 1 биткойн, а по другой транзакции на этот же адрес поступило 2 биткойна, третья транзакция отправила с этого адреса 1 биткойн. Но в базе не хранится отдельной записи, сколько всего сейчас биткойнов числится за данным адресом — просто предоставляется возможность в любой момент это легко подсчитать. Такие подсчёты автоматически делают клиентские программы, пользователь может и не замечать раздробленности информации.

Каждый пользователь системы может генерировать неограниченное количество пар ключей (алгоритм ECDSA с параметром secp256k1). Размер закрытого ключа — 256 бит, а соответствующего ему открытого ключа — 512 бит.

Основное использование ключей — создание биткойн-адреса и подтверждение правомочности формирования транзакций. Но они могут использоваться и для цифровой подписи или шифрования.

Создание новой пары ключей автономно и не требует соединения с сетью или Интернетом, так как создание одинаковых ключей очень маловероятно.

Адреса создаются при помощи генерации асимметричной пары криптографических ключей, для чего не требуется подключения к Интернету. Человек может иметь неограниченное число адресов, создавая их по своему желанию. Все адреса записаны в децентрализованную цепочку блоков транзакций, защищённую от изменений.

Биткойн-адрес является последовательностью байт, полученных в результате преобразования открытого ключа. Чаще всего кодированием Base58 адрес записывают как строку длиной до 34 букв латинского алфавита и цифр.

Передача биткойнов на адрес происходит с контролем формальной корректности адреса получателя, но без контроля реального существования ключа, который бы ему соответствовал.

Транзакции включают в себя открытый ключ, хэш пердыдущей транзакции и цифровую подпись отправителя. На рисунке 1 показана упрощённая структура последовательных транзакций с одним входом и одним выходом.

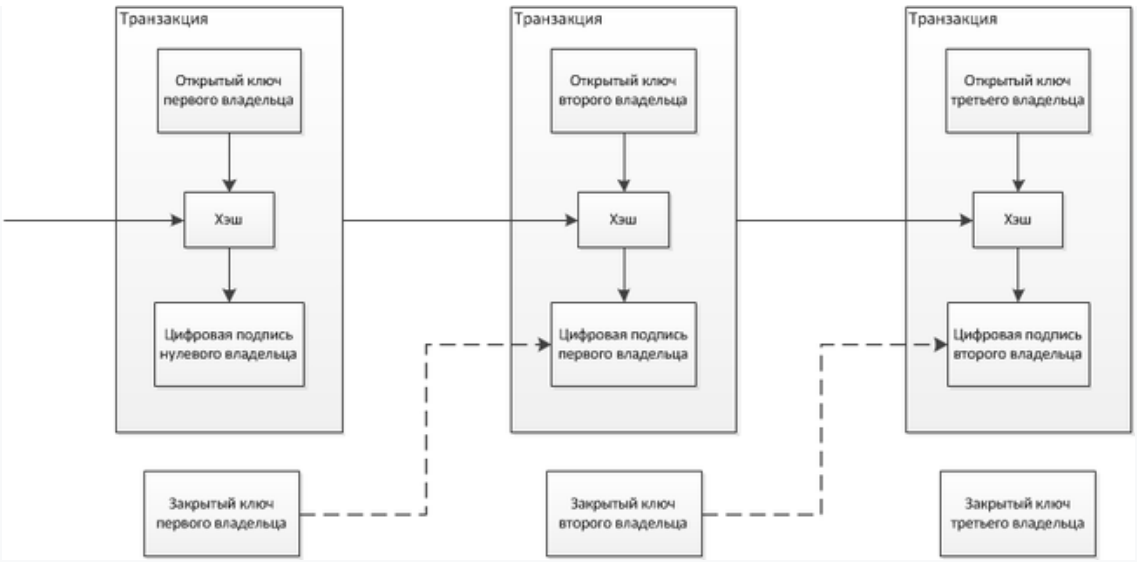

Рисунок 1. Упрощённая структура последовательных транзакций с одним входом и одним выходом.

Транзакция записанная в блок Биткоина оперирует такими понятиями, как **Входы** (Inputs или In) и **Выходы** (Outputs или Out).

Входы (In)—это транзакции, пополняющие биткоин-адрес, а **Выходы** (Out)—это суммы, которые переводятся на другие биткоин-адреса.

Поэтому в Биткоине новая транзакция через Входы (один или несколько) ссылается на Выходы (один или несколько) предыдущих транзакций и формирует Выходы (также один или

несколько) для использования в последующих транзакциях. На рисунке 2 показана структурная схема входов и выходов в нутрии транзакций.

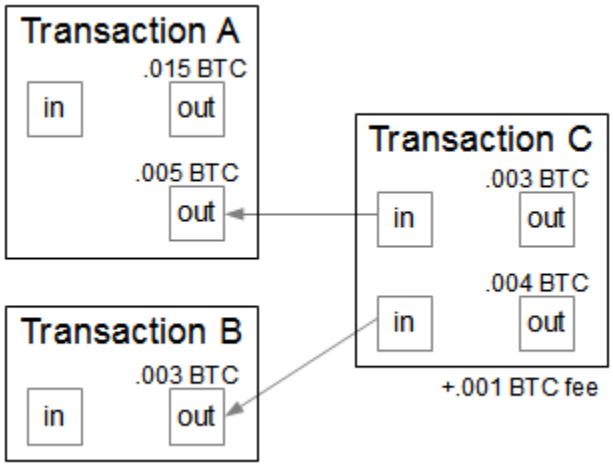

Рисунок 2. Входы и выходы транзакций

Структура записи транзакций в блокчейне Биткоина содержит:

– количество используемых Входов;
– хэш-код и индекс каждого Входа, а также служебная информация;
– количество используемых Выходов;
– суммы Выходов, а также служебная информация;
– метка времени транзакции.

Размер транзакции. Каждый адрес, с которого получены средства — это ±148 байтов. Каждый адрес, на который уходят средства (минимум 2, т.к, отправляешь сдачу себе) — это ±34 байта. Каждая транзакция занимает еще ±10 байтов, независимо от количества адресов, которые в ней участвуют. Получаем, что в среднем 226 байта уходит на 1 транзакцию.

Сеть работает по следующим правилам:

1) Новые транзакции рассылаются всем узлам.
2) Каждый узел объединяет пришедшие транзакции в блок.
3) Каждый узел пытается подобрать хэш блока, удовлетворяющий текущей сложности.

4) Как только такой хэш найден, этот блок отправляется в сеть.

5) Узлы принимают блок, только если все транзакции в нем корректны и не используют уже потраченные средства.

6) Свое согласие с новыми данными узлы выражают, начиная работу над следующим блоком и используя хэш предыдущего в качестве новых исходных данных.

Участники всегда считают истинной самую длинную версию цепочки и работают над ее удлинением. Если два узла одновременно опубликуют разные версии очередного блока, то кто-то из остальных пиров получит раньше одну версию, а кто-то — другую. В таком случае каждый начнет работать над своей версией цепочки, сохранив другую на случай, если она окажется продолжена раньше. Двойственность исчезнет, как только будет получен новый блок, который продолжит любую из ветвей, и те узлы, что работали над конкурирующей версией, переключатся на нее. Новые транзакции не обязательно должны достигать всех узлов. Если о них будет знать достаточно много узлов, вскоре они попадут в один из блоков. Правила рассылки блоков тоже не являются строгими в отношении потерянных сообщений. Как только узел, пропустивший один из блоков, получит уже следующий за ним, он запросит недостающую информацию, чтобы заполнить очевидный пропуск.

На рисунке 3 приведена упрощенная структурная схема блоков, включающая в себя хэш предыдущего блока, хэш транзакций и добавочное число, служащее для регулирования сложности нахождения блоков.

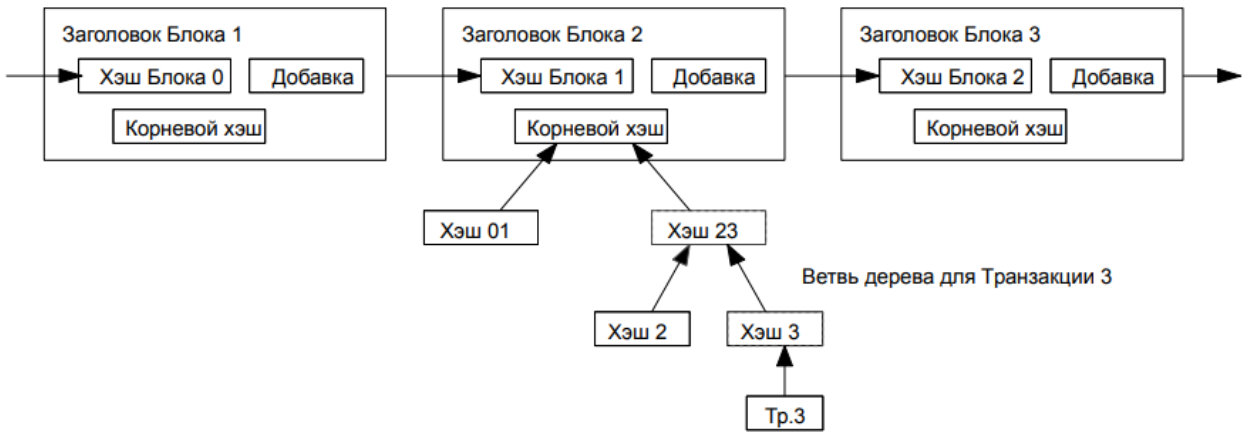

Рисунок 3. Структурная схема Блоков

Каждый блок всегда содержит свой порядковый номер и хеш предыдущего блока. Все блоки можно выстроить в одну цепочку, которая содержит информацию о всех совершённых когда-либо операциях с биткойнами.

Параметр «сложность» каждые 2016 блоков (примерно раз в две недели) автоматически устанавливается так, чтобы поддерживать постоянной среднюю скорость создания блоков (примерно 1 блок в 10 минут). Если блоки формируются быстрее, то после пересчёта «сложности» достичь цели становится труднее, и наоборот. Поэтому изменение суммарной вычислительной мощности сети лишь очень незначительно изменяет количество создаваемых блоков.

До включения транзакции в блок есть техническая возможность оформления нескольких разных транзакций по передаче с одного адреса одних и тех же биткойнов разным получателям. Как только транзакция будет включена в блок, остальные транзакции с этими же биткойнами система будет уже игнорировать, то есть в цепочке блоков останется только одна транзакция. Но если контролировать более 50 % суммарной вычислительной мощности сети, то существует теоретическая возможность при любом пороге подтверждений формировать параллельную более длинную цепочку блоков, в которой те же биткойны будут переданы другому получателю (проблема «двойного расходования»). Когда сеть получит сведения о второй цепочке блоков, она станет основной, а транзакция в ней — подтверждённой, первая же транзакция утратит подтверждения и будет считаться ошибочной. В результате не произойдёт удвоения биткойнов, но изменится их текущий владелец, при этом первый получатель утратит биткойны без каких-либо компенсаций.

Способы масштабирования сети

Проблема масштабируемости биткойна связана с изначальным ограничением разработчиками размером в один мегабайт базовой структуры для хранения данных(блока) в его блокчейне. Такое ограничение обусловлено особенностью построения блокчейна, как полностью реплицированной распределённой базы данных, что требует постоянной пересылки между всеми участниками каждого нового элемента. Уменьшение размера блока существенно ограничивает эффективность потенциальной DDoS-атаки. Если учесть среднее время на формирование блока (10 минут) и средний размер информации о транзакции(3бита), то для гарантированного помещения в блок(1Мб) количество транзакций не должно быть слишком велико — на уровне около трех транзакций в секунду.

С ростом популярности сети Биткойна, число транзакций увеличилось, но из-за ограничения максимального размера блоков не все транзакции «помещались» сразу, периодически возникала очередь. В мае 2017 года ситуация сильно ухудшилась, ожидание включения транзакции в блок достигало нескольких дней.

На рисунке 4 изображена цепочка блоков , в которой, основная последовательность блоков (чёрные) является самой длинной от начального (зелёный) до текущего. Форки порождают побочные ветви (фиолетовые), которые впоследствии отсекаются.

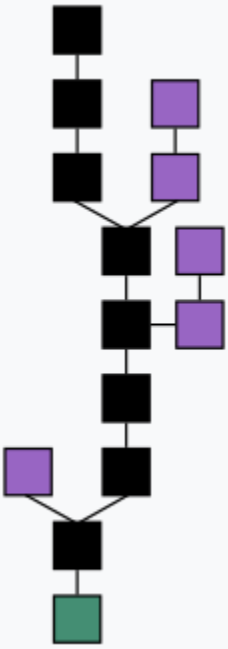

Рисунок 4. Цепочка блоков сети Блокчейн.

Форк блокчейна — деление непрерывной цепи на две ветви. Такое деление в сети биткойн является частью алгоритма и происходит регулярно в процессе майнинга. Новые блоки всегда содержат ссылку на предшественника. Если два блока, сформированных разными майнерами, ссылаются на один и тот же блок-предшественник, это и есть деление. Система не даёт автоматического предпочтения. Новые блоки в качестве предшественника могут указать любого из них. В результате на некоторое время сеть разделяется. В какой-то момент одна из веток станет длиннее и «отставшая» отмирает, так как система истинной считает самую длинную цепь.

Блокчейн также может разделяться, когда разработчики изменяют протокол определения того, какие блоки действительны.

«*Хардфорком*» называют разбиение блокчейна на две отдельные цепи в результате использования двух разных протоколов. Новый протокол также может разделить сеть, если все участники сети не следуют за ним. Например, Bitcoin Cash возник в результате «хардфорка» Bitcoin, для увеличения пропускной способности транзакций в секунду, путем увеличения обьема блока до 10Мб.

Сообщества Bitcoin XT, Bitcoin Classic и Bitcoin Unlimited предложили увеличение предела размера блока, как способ увеличения масштабируемости. Однако поддержка обоих предложений со временем падала.

В отличие от «хардфорка», «софтфорк» — это изменение протокола, в результате которого созданные блоки признаются действительными и старым программным обеспечением. Согласно *Investopedia*, «софтфорк» также может разделить сеть, если необновлённое программное обеспечение будет создавать блоки, которые не будут считаться действительными по новому протоколу.

SegWit:

- Меняет структуру хранения данных в каждом блоке биткойна.
- Обеспечивает повышение пропускной способности транзакции, оставаясь совместимым с более ранними версиями программного обеспечения биткойна.
- Устраняет транзакционную инертность, которая стала препятствием для других проектов биткойнов.
- Внедрение Lightning Network стало осуществимым.

Lightning Network —целью проекта является устранение проблемы масштабируемости биткойна путём масштабирования «вне сети». Он предназначен для обеспечения обновления состояния микроканала без использования каких-либо блокировок (в обычном несостязательном случае), что делает микроплатежи. Lightning Network потребует, чтобы транзакция финансирования на блокчейне открыла канал

Сравнение пропускной способности сети Bitcon и Ethereum

Пропускная способность сети блокчейн считается в количестве обработанных транзакций в нутрии сети за одну секунду. Для этого необходимо узнать эти показатели для Bitcoin и Ethereum. Будем считать средние показатели для каждой сети.

В таблице 1 приведены параметры для каждой из сети на 20.05.2018. Рассматривались показатели за весь день, чтобы усреднить данные.

Таблица 1

Параметры сетей

	Размер Блока (kb)	Вреимя создания блока (м)	Количество Транзакций за день
Bitcoin	457.227	10	158.519
Ethereum	23.759	0.153	777.283

Данные для таблиц были взяты из дневных графиков.

На рисунке 5 показан график изменения размера блоков в течении месяца.

Рисунок 5. Средний размер блока

Как мы видим у сети Bitcoin показатель около 457 килобайт, а у сети Ethereum это 23 килобайта за один блок.

Средняя скорость создания блоков этих сетей показана на рисунке 6.

Рисунок 6. Среднее время создания блока

Число транзакций за день на рисунке 7.

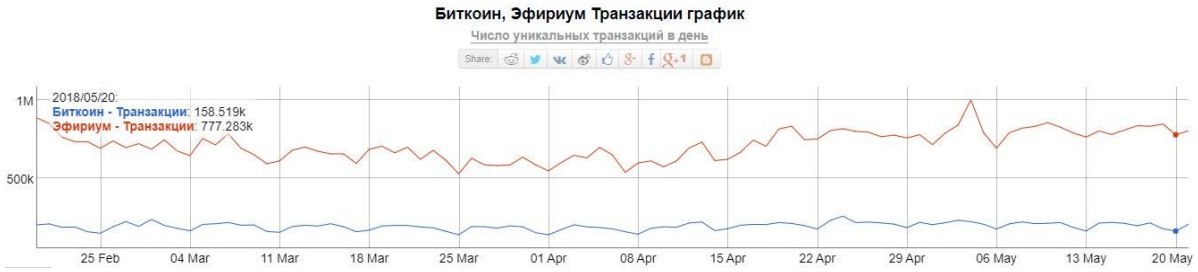

Рисунок 7. Число транзакций за день

Теперь рассчитаем максимальную пропускную способность для каждой сети.

Bitcoin – Максимальный размер блока 1 Мбайт, средний размер транзакции 224 байт, время создания блока 10 минут. Значит сеть Bitcoin за секунду может обрабатывать до 7 транзакций.

Etereum – Размер блока в этой сети определяется условными единицами, называемыми газом. Вместимость одного блока составляет 4 723 357 газа, а для отправки одно транзакции требуется от 21 000 газа, значит, в блок помещается где-то 224 транзакции. Скорость формирования блоков от 15 – 20 секунды. Значит в секунду сеть Ethereum может обрабатывать до 15 транзакций.

Библиографический список

1. Сеть [Электронный ресурс] : / URL : https://en.bitcoin.it/wiki/Network .
2. Алекс Талскотт, Дон Талскотт. Технология Блокчейн. - М.: Моск., 2017.
3. Принципы работы сети биткоин [Электронный ресурс] : / URL : https://ru.bitcoinwiki.org/wiki/ .

СЕКЦИЯ 3. ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

УДК 004

Бариев И.И., Гильфанов К.Х. Поиск альтернативных методов при использовании Unity 5

Searching for alternative methods when using Unity 5

Бариев И.И.
магистр, Казанский национальный исследовательский технологический университет, РФ, г. Казань
Гильфанов К.Х.
Научный руководитель,
доктор технических наук, профессор, Казанский государственный энергетический университет, РФ, г. Казань
Bariev I.I.
Master's degree, Kazan National Research Technological University, Russian Federation, Kazan
Gilfanov K.Kh.
Scientific adviser,
Doctor of Technical Sciences, Professor, Kazan State Power Engineering University, Russian Federation, Kazan

Аннотация. *В процессе работы в сторонних программах программист работающий ранее в компиляторах предназначенных только для его языка программирования может столкнуться с проблемой, связанной с тем , что некоторые сторонние библиотеки (и даже некоторые стандартные) некорректно отображаются (или полностью отсутствуют) в данной среде программирования. Причина этого может быть в том , что в других средах программирования могут быть свои методы и функции заменяющие их и тот метод который использовали программисты при работе с компиляторами более не подходит. Основная цель работы – Показать программистам ,работающим только с компиляторами, какой компилятор стоит в Unity 5, и где брать библиотеки заменяющие стандартные библиотеки языка C#.*

Ключевые слова*: Unity 5, C# для Unity, метод Parse, метод ConvertTo.*

Abstract*. In the process of working in third-party programs, the programmer working earlier in compilers intended only for his programming language may encounter a problem due to the fact that some third-party libraries (and even some standard ones) are incorrectly displayed (or completely absent) in this programming environment. The reason for this may be that other programming environments can have their own methods and functions replacing them and the method that programmers used when working with compilers is no longer suitable. The main goal of the work is to show programmers working only with compilers, what compiler is in Unity 5, and where to take libraries that replace standard libraries of C #.*

Keywords*: Unity 5, C # for Unity, Parse method, ConvertTo method.*

Современный разработчик C# умело оперирует такой библиотекой как https://msdn.microsoft.com. В ней содержаться все опредение всех методов, принимаемых значений,аргументов и перегрузок. Именно оттуда большинство программистов и берут методы для работы с той или иной программой написанной на языке C#. Однако в ней нет некоторых вещей , применимых к миру 3D моделирования. Нет функций использования физических движков , систем столкновений , и расчета масс для какого то конкретного физического предмета. Поэтому, многие программисты , игростроители и специалисты по виртуальной реальности исполььзую дополненый язык –C# для Unity.

При переходе на новую платформу я очень долго искал таковую. В первую очередь выискивал для себя уже готовую технологическую платформу которую не надо будет дополнительно настраивать и подгонять. И основными критериями для меня были адекватная цена и по возможности — мультиплатформенность, чтобы один раз написанный код можно было использовать снова без перевода из одного языка в другой . Почти сразу я наткнулся на Unity 5 .

Ключевыми плюсами для меня во время использования Unity 5 да и вообще и C# в частности стали :

- Использование .NET Framework и C# (лично для меня это очень удобно, особенно интеграция в Visual Studio)
- Возможность сделать сборку сразу на Android , iPhone и на другие мобильные устройста и для веб-плеера (standalone сборки меня не прельщают)
- Уже готовые сценарии поведения на C# можно будет еще раз использовать, например, при создании портативной вресии которую можно выпустить на другие платформы)
- Удобный интерфейс и не слишком прожорливые редакторы, едят они очень мало)
- Приемлимая цена(Основной движок вообще бесплатный, платные только дополнения)

Основные ссылки и источники по которым можно найти ответы на многие свои вопросы:

- https://docs.unity3d.com Библиотека Unity подобная библиотеке Msdn в .NET. Так же содержит определения всех скриптов, методов и перегрузок методов.
- Unity Answers. Ответы на все популярные вопросы, оссобенно у новичков.
- UnifyWiki. Можно декомпилировать код и посмотреть результат (он не обфусцирован).
- Форумы answers.unity3d.com и forum.unity3d.com. Также я читал книгу Game Engine Architecture Джейсона Грегори.
-

- Что надо иметь ввиду при переходе на Unty:
- Каждый отдельный скрипт — это класс, описывающий поведение игровой сущности.
- Unity не переваривает namespace. В будущем обещают исправить, но пока так.

• Имя класса и имя файла скрипта должны совпадать. Если вы создадите класс через Visual Studio, то не забудьте брать namespace. И наследовать класс от MonoBehaviour (using тоже не забудьте). Если вы создадите класс через Unity, то по умолчанию он будет называться NewBehaviourScript, не забудьте переименовать как файл скрипта, так и имя класса.

• вы хотите, чтобы значения параметров можно было менять в игровом редакторе? Тогда делайте их публичными.

• Не путайте класс CharacterController и его тип typeof(CharacterController)

• инструкции типа "@script RequireComponent(CharacterController)" превращаются в аттрибуты класса — [RequireComponent(typeof(CharacterController))]

• Плавающая запятая представлена типом float, на double даже вектор не помножить =)

• В js не надо было явно приводить типы. C# — не такой!

• И если вам нужно что-то сложное инициализировать, воспользуйтесь готовыми функциями Awake() и Start(), перегружать конструктор MonoBehaviour не рекомендуется.

Так же учитывайте что в Unity есть 3 независимые друг от друга ветви событий:

• События, вызываемые по событиям масло масляное (загрузка сцены, выход пользователя).

• События, вызываемые при прорисовке кадра. В этом случае все используемые скрипты вызываются в цикле прорисовки экрана, а значит, будут непосредственно влиять на FPS (частоту кадров в секунду). Поэтому здесь нужно очень аккуратно работать с функциями, которые требуют много времени на обработку.

• События, вызываемые при расчёте физики. Для расчёта физики создаётся отдельная, независимая нить, события в которой вызываются с определённым интервалом времени. Размер этого интервала можно настроить в пункте меню: Edit -> Project Settings -> Time -> Fixed Timestep.

Так же помимо основных ветвей можно создавать дополнительные ветви –сопрограммы.

Короутины (Coroutines, сопрограммы) в Unity —способ запускать функции, которые должны работать параллельно в течение некоторого времени. То есть вам не обязательно знать , что за чем идет, является ли функция физической или логической. В работе с короутинами ничего принципиально сложного нет, и интернет полон статей с поверхностным описанием их работы. Однако, нигде не сказаны конкретные функции их работы.

Хочу предложить вам небольшой паттерн, реализующий такую возможность, а также подбор информации о короутинах.

Короутины представляют собой простые C# итераторы, возвращающие IEnumerator и использующие ключевое слово yield. В Unity короутины регистрируются, и выполняются до первого yield с помощью метода StartCoroutine. Дальше Unity вызывает метод который опрашивает зарегистрированные короутины после каждого вызова Update и перед вызовом

LateUpdate, определяя по возвращаемому в yield значению, когда нужно переходить к следующему блоку кода и условие по которому нужно ощутить этот переход.

Существует несколько вариантов для возвращаемых в yield значений:

Обычно вызов сопрограммы выполняется после возвращения функции Update(). Сопрограмма это функция, которая может приостановить исполнение (yield), пока не будет выполнена. Различные виды использования сопрограмм:

yield: сопрограмма будет продолжена после всех функций Update(), которые будут вызваны в следующем кадре.

yield WaitForSeconds(2): Продолжить после указанного времени задержки, когда все функции Update() уже были вызваны в кадре

yield WaitForFixedUpdate(): Продолжается, когда все функции FixedUpdate() уже были вызваны

yield WWW: Продолжается, когда загрузка WWW-контента завершена.

yield StartCoroutine(MyFunc): Связи сопрограмм, вызов сопрограммы будет ожидать завершения функции MyFunc.Выйти из короутина можно следующим образом : yield return break;

Небольшой пример с применением короутинов:

Суть работы кода такова: текстура на экране двигается по фигуре "крендель" с равной скоростью. Ничего особенного, и я даже пример выбрал не особо показательный, но все же продолжу.

Короутина позволяет прерывать вычисления функции и продолжать с того же места, на котором остановились. В моем примере есть функция Coroutine, которая имеет бесконечный цикл, из которого нет выхода. Каждая итерация цикла считает новое положение текстуры на экране и останавливается на 10 мс, перед тем как продолжить. Но загадка в том, что прерывание этой функции происходит не с помощью остановки процесса, а с помощью механизма короутин юнити.

Итак последовательно:

1. При старте скрипта запускается короутина (функция с названием Coroutine).

2. Эта функция выполняется до строчки yield return new WaitForSeconds(0.01f); которая прерывает ее работу на 10 миллисекунд.

3. Прерывание не влияет на остальные функции (процесс не останавливается и действие происходит в том же потоке)

4. По истечении времени короутина продолжает бесконечный цикл пока снова не доходит до строки с задержкой.

5. Функция OnGUI рисует текстуру на экране.

Возможные прерывания выполнения кода:

1. yield return null; - прерывает выполнение короутины до следующего кадра.

2. yield break; завершает выполнение короутины.

3. yield return new WaitForSeconds(количество секунд); - прерывает на время.

4. yield return new WaitForEndOfFrame(); - прерывает выполнение до конца кадра.

5. yield return new WaitForFixedUpdate(); - прерывает выполнение до кадра, в котором обновляется физика.

При использовании функции WaitForSeconds инициализируется непрерывно существующий объект в памяти , поэтому его использование в быстрых циклах может быть плохой идеей.

Уже было сказано, что короутины работают параллельно, следует уточнить, что они работают не асинхронно, то есть выполняются в том же потоке.

Следует обратить внимание на то, что в короутине сначала вызывается функция yield return null, и только потом идет запись в лог. В общем случае это имеет значение, потому что выполнение короутина начинается в момент вызова функции StartCoroutine(TestCoroutine()), а переход к следующему блоку кода (yield return nul)l будет осуществлён после метода Update. Так что Time.deltaTime будет указывать на одно и то же значение.

Такж, что короутины с бесконечным циклом всё еще можно прервать, вызвав StopAllCoroutines(), StopCoroutine(«TestCoroutine»), или уничтожив родительский Object.

Хорошо. Значит с помощью короутинов мы можем создавать триггеры, проверяющие определенные значения каждый фрейм, можем создать последовательность запускаемых друг за другом короутинов, к примеру, проигрывание серии анимаций, с различными вычислениями на разных этапах. Или просто запускать внутри короутина другие короутины без yield return и продолжать выполнение.

Во многих случаех приходится прибегать и к Синглтонам.

Во-первых, то, что такое Синглтон? Это шаблон проектирования, который ограничивает создание экземпляра класса одним объектом. И, если вы здесь, вы, вероятно, хотите использовать это в основном для реализации глобальных переменных. Для любого другого использования, начав отсюда вы не проиграете.

Преимущество использования синглетонов в Unity, а не статических параметров и методов, в основном:

(1) статические классы загружается при первом обращении к ней, но, должен быть, пустой статический конструктор (или он будет создан). Это означает, что легче испортить и сломать код, если Вы не будете осторожны и не знаете, что делаете.Т.е. работать со статическими классами может только опытный программист. Что касается использования одноэлементного шаблона, Вы автоматически уже делаете много ненужных вещей, таких как создание их с помощью статического метода инициализации и приравнивания его к неизменяемому.

(2) Синглтон может реализовать интерфейс (не статический). Это позволяет создавать контакты, которые можно использовать для других синглтонов или просто для любого другого класса, который вы хотите использовать для этого . Другими словами, Вы можете иметь объект программы с другими компонентами на нем для лучшей организации!

(3) Вы можете также наследовать данные от базовых классов, которые Вы не можете передать через не статический класс.

П. С.: к сожалению, нет хорошего способа, чтобы удалить при необходимости "экземпляр класса" используя синглтон..

Какие цели лично я преследую при использовании синглтона?

- Во-первых – легкий доступ к экземпляру класса в коде любого компонента.
- Во-вторых – запрет на добавление нескольких компонентов в сцену, если по логике этот компонент должен быть уникальным (настройки, BestScore.....). Причем запрет должен действовать и на уровне редактора, и на уровне кода.
- В-третьих – все должно быть понятно, поведение компонента естественно и ожидаемо, логика синглтона не должна мешать нормальной работе компонента и не должна сказываться на производительности готового приложения.

1) При пересборке проекта (читай – при ЛЮБОМ изменении в любой части кода) коллекция _instances отчищается, т.к. класс UniqueComponentAtScene не является сериализуемым. А быть он таким не может, потому что нам заранее неизвестно, насколько сложные будут наследующие классы. Возможно их нельзя будет сделать сериализуемыми в принципе. И «ближайший» вызов метода Awake, где _instances заполняется, произойдет только в случае запуска проекта в редакторе.

Т.е. поведение компонента UniqueComponentAtScene – НЕ ОЖИДАЕМО, и заставляет каждый раз жмакать кнопку Play в редакторе. Можно забыть про это и словить внезапную ошибку NullReferenceException.

2) Если наследующий UniqueComponentAtScene класс1 будет иметь атрибут RequireComponent с указанием на класс2, который тоже является наследником UniqueComponentAtScene – то произойдет ошибка при добавлении дубля компонента класса1 на сцену.

Для решения первой проблемы я использую такой хитрый атрибут, как DidReloadScripts. Упоминаний про него я нигде не встречал и вообще наткнулся совершенно случайно. Он позволяет пометить статичный метод, который будет вызван в редакторе после пересборки

проекта или при первой загрузке редактора. После этого нам надо найти всех наследников UniqueComponentAtScene с помощью рефлексии и найти экземпляры этих классов на сцене. Когда все экземпляры, или компоненты, или синглтоны, найдены – вызываем для них Awake.

Правда здесь тоже есть нюанс – почему-то метод, расположенный в абстрактном генерик классе и помеченный атрибутом DidReloadScripts, вызывает ошибку "TypeLoadException: A type load exception has occurred". Толи баг, толи моих знаний языка не хватает для понимания... В общем, необходимо добавить прямо в файл с кодом UniqueComponentAtScene скриптуемый объект, в котором уже прописать нужный метод с атрибутом DidReloadScripts.

Библиографический список

1. Сопрограмма https://ru.wikipedia.org/wiki/Сопрограмма
2. Unity в действии. Мультиплатформенная разработка на C# https://www.manning.com/books/unity-in-action

УДК 004

Бариев И.И., Гильфанов К.Х. Порядок формирования отображения в виртуальном пространстве

The order of formation of a mapping in a virtual space

Бариев И.И.
магистр, Казанский национальный исследовательский технологический университет, РФ, г. Казань
Гильфанов К.Х.
Научный руководитель,
доктор технических наук, профессор, Казанский государственный энергетический университет, РФ, г. Казань
Bariev I.I.
Master's degree, Kazan National Research Technological University, Russian Federation, Kazan
Gilfanov K.Kh..
Scientific adviser,
Doctor of Technical Sciences, Professor, Kazan State Power Engineering University, Russian Federation, Kazan

Аннотация. *На сегодняшний день широта использования информационных технологий выходит на новый уровень, если раньше инновационные решения в области информатики использовались только в науке, то сегодня повсеместное внедрение информационных технологий во все сферы жизни человека. Основная цель данной работы- разъяснить специалистам впервые работающим в среде виртуальной реальности, в какой последовательности происходит расчет тех или иных показателей . В данной работе будет рассмотрен алгоритм работы виртуальной среды Unity 5.*

Ключевые слова*: Unity 5, виртуальная система, алгоритм работы виртуальной системы.*

Abstract. *To date, the breadth of the use of information technology is reaching a new level, if earlier innovative solutions in the field of informatics were used only in science, today the widespread introduction of information technologies in all spheres of human life. The main goal of this work is to explain to specialists who work for the first time in a virtual reality environment, in which sequence the calculation of certain indicators takes place. In this paper, we will consider the algorithm of the Unity 5 virtual environment.*

Keywords: *Unity 5, virtual system, algorithm of virtual system operation.*

Ниже представлен полный алгоритм вычислений который используется при просчете каждого кадра в среде виртуальной разработки Unity:

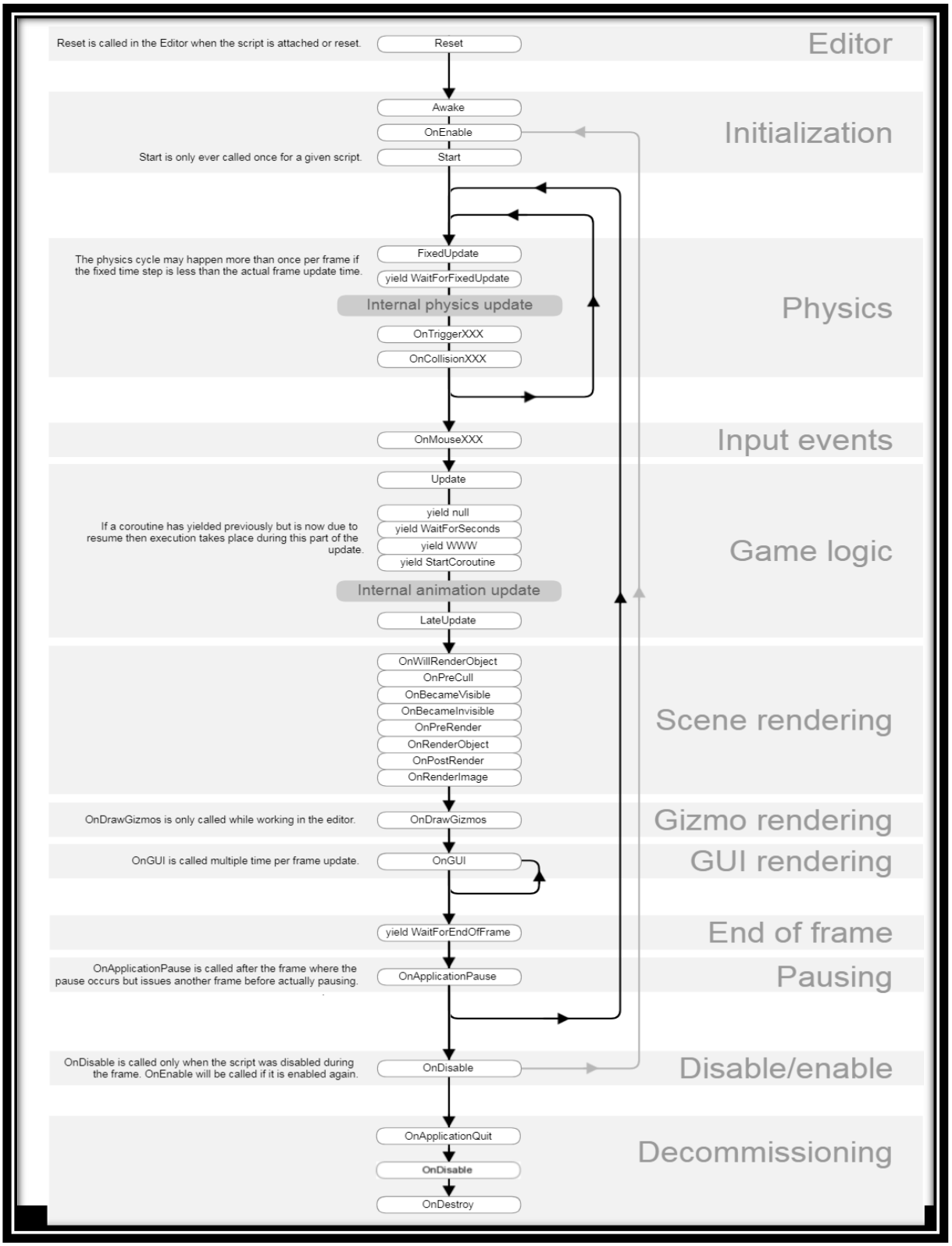

Рисунок 1. Жизненный цикл скрипта в программе Unity 3D

Краткое разъяснение схемы (Рис.1):

Первой функцией, которая отображается графическим движком является функция Awake. Она инициализирует все запрошенные исходники, ссылки на материалы, префабы и т.д. Следующей функцией включается функция Start.Она отвечает за начальное расположение персонажа, управляемого оператором, расположение вещей, окружения и условий мира. Следом инициализируется функция FixedUpdate. Она проверяет соответствие физическим законам, гравитации ,систем столкновений физических объектов и т.д. Следующая функция Update. Основная функция обновления кадров, именно она понимает и принимает все взаимодействия пользователя через управление, и отвечает пользователю взаимодействием. Следом идет LateUpdate. Сюда входят события или методы которые должны отреагировать на движения пользователя в предыдущем методе. Следующая функция OnTriger. Вызывается и обрабатывает вхождения заданных объектов в некоторое поле при заданных условиях. Допустим, если пользователь зашел в помещение, а у нее есть некоторое поле то произойдет некоторое событие описанное ниже (Например, появится возможность взаимодействовать с окружением).Функция OnCollizion. Вызывается при столкновении одного физического объекта с другим. Часто используется при имитации взаимодействия взрывоопасных веществ с пользователем. Функция OnDrawGizmos вызывается при выводе каких то показателей на экран (температура или погрешность на приборе). Функция OnApplicationPause . Используется для возможности поставить программу на паузу и выхода в меню настроек. Функция OnDisable вызывается при отключении каких то объектов , эффектов ,и явлений. Функция EndOfFrame вызывается при конце кадра и является завершающей функцией, после которой больше не ведется расчет и начинается расчет следующего кадра.

В основе своей программистами используются только некоторые функции из данного списка. Однако, лучше знать, какие функции используются до и после твоих действий что б иметь возможность на них повлиять.

Библиографический список

1. Отличный курс уроков по Unity3D и C# От команды Ravian Team http://www.unity3d.ru/distribution/viewtopic.php?f=11&t=22117
2. C# ПОЛНЫЙ КУРС https://proglib.io/p/gamedev-compilation/

СЕКЦИЯ 4. СТРОИТЕЛЬСТВО И АРХИТЕКТУРА

УДК 625.768.5

Сопец В.В. Виды зимней скользкости и возможности их определения с помощью дорожных датчиков

Types of winter slipperiness, and the possibility of their determination by using road sensors

Сопец Владислав Владимирович,
Воронежский государственный технический университет
Научный руководитель
Самодурова Т.В. , д.т.н., профессор кафедры Проектирования автомобильных дорог и мостов
Воронежский государственный технический университет
Sopets Vladislav Vladimirovich
Voronezh State Technical University
Scientific adviser: Samodurova T., Doctor of Engineering, Professor of road and bridge design Department,
Voronezh State Technical University

Аннотация*. Рассмотрены классификации видов зимней скользкости, образующихся на дорожных покрытиях. Приведен сравнительный анализ дорожных датчиков, используемых в системах дорожного погодного мониторинга. Для оснащения дорожных метеосистем рекомендован бесконтактный датчик.*
Ключевые слова*: автомобильная дорога, зимняя скользкость, дорожный погодный мониторинг, дорожные датчики*
Abstract. *The classification of types of winter slippery formed on road surfaces is considered. A comparative analysis of road sensors used in road weather monitoring systems is presented. A non-contact sensor is recommended for equipping road meteorological systems.*
Keywords: *road, winter slippery, road weather monitoring, road sensors*

Для повышения безопасности движения в сложных погодных условиях (снегопады, метели, гололед, туманы и т.д.) предусмотрен контроль погоды на автодорогах (погодный мониторинг). Основная цель погодного мониторинга - получение информации для проведения работ, позволяющих сохранять и восстанавливать в заданные сроки сцепные качества дорожных покрытий путем ликвидации или профилактики образования гололедных и снежных отложений. Проведение таких работ обеспечивает безопасность движения в сложных погодных условиях [1].

К зимней скользкости относят все виды снежно-ледяных отложений, снижающие сцепление искусственного покрытия с покрышками колес транспортных средств. Существует несколько классификаций гололедно-изморозевых явлений и различных видов зимней

скользкости дорожных и аэродромных покрытий, в основу которых положены определенные понятия: процессы образования, структура отложений, физические свойства, технологи работ по ликвидации и т.д. В нормативно-технических документах используется простая классификация, в основу которой положено довольно четкое различие видов зимней скользкости как по внешним признакам, так и по физическим свойствам. Эта классификация удобна для организации работ по борьбе с зимней скользкостью, так как каждый ее вид легко определяется визуально: рыхлый снег, уплотненный снег и стекловидный лед. Однако она ориентирована на технологии работ, в основе которых лежит ликвидация уже образовавшейся скользкости, а не ее профилактика [2].

В зависимости от условий образования, которые позволяют прогнозировать возникновение скользкости, Т.В. Самодуровой предложено выделить пять видов зимней скользкости [3,4].

Классификация различных видов зимней скользкости дорожных покрытий и условия их образования представлены в таблице 1.

Таблица 1

Классификация различных видов зимней скользкости дорожных покрытий и условия их образования

Вид зимней скользкости	Условия образования			
	Температура воздуха	Температура покрытия	Осадки, их вид	Состояние покрытия
Гололедица	Ниже 0°С	Ниже 0°С	Любые, выпадающие при температуре воздуха выше -3°С	Мокрое
Черный лед	То же	Ниже 0°С Ниже точки росы	Нет	Сухое
Гололед и твердый налет	Выше 0°С От 0 до -5°С	Ниже 0°С То же	Жидкие или переохлажденные Мокрый снег	- -
Снежный накат	От 2 до 0°С От 0 до -6°С От -6 до -10°С	- - -	Твердые (снег, мокрый снег) То же То же	- - -
Рыхлый снег	От -6 до -10°С Ниже -10°С	- -	То же То же	- -

Исследования, проведенные в ВГТУ, позволили получить данные о количестве обработок дорожного покрытия в зимний период при образовании различных видов зимней скользкости [4]. Результаты исследований для дорог Черноземья приведены на диаграмме - рисунок 1.

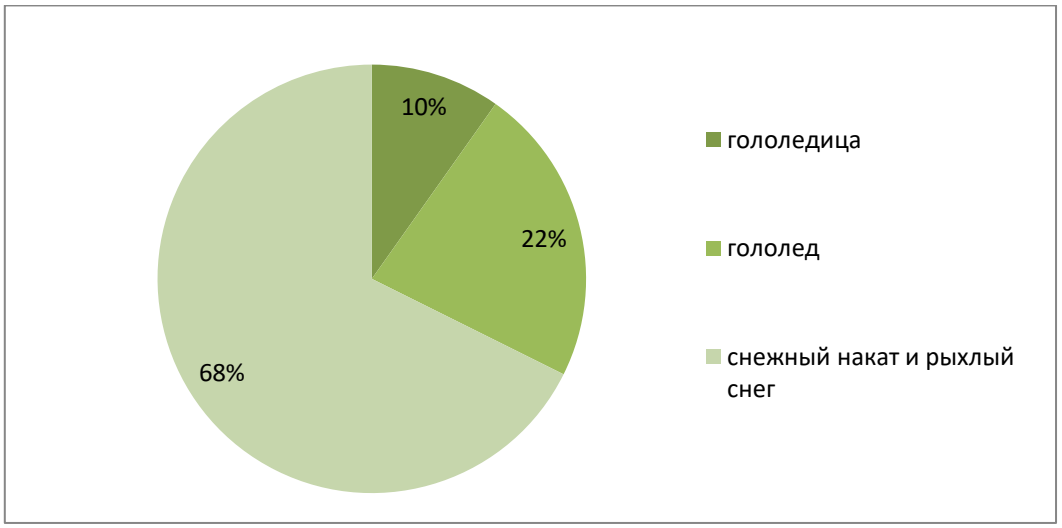

Рисунок 1. Диаграмма распределения количества обработок дорожного покрытия

Анализ диаграммы показывает, что наиболее часто (в 68 %) на дорожном покрытии в центральной части России образуется снежный накат или слой рыхлого снега.

Для организации работ по борьбе с зимней скользкостью и сбора информации о погодных и дорожных условиях предусматривается установка вдоль дорог автоматических дорожных метеорологических станций (АДМС).

АДМС – комплект датчиков, расположенных непосредственно у дороги и предназначенных для сбора метеорологической и дорожной информации и передачи ее в центр сбора, обработки и хранения.

Наиболее ценным по информативности в АДМС является дорожный датчик. Информация, полученная с дорожных датчиков, позволяет оценить температуру поверхности и состояние дорожного покрытия в режиме реального времени. Все известные к настоящему времени дорожные датчики можно разделить на четыре основных типа: активные, пассивные, дистанционные (бесконтактные) и мобильные.

Сравнительная характеристика дорожных датчиков с указанием их достоинств и недостатков приведена в таблице 2.

Пассивный датчик измеряет существующую температуру дорожного покрытия или температуру на определенной глубине в дорожной конструкции. Он определяет также состояние покрытия – сухое, мокрое, скользкое и измеряет концентрацию противогололедных реагентов. Фактически он фиксирует состояние дорожного покрытия, но не позволяет прогнозировать его изменение.

Активный датчик позволяет прогнозировать возможное образование скользкости на покрытии и время ее образования. Для этого в конструкции датчика предусмотрена возможность охлаждения его поверхности на 2°С по отношению к измеренной температуре

дорожного покрытия. Если на поверхности охлажденного датчика образуется лед или иней, то зная, динамику понижения температуры воздуха можно определить время обледенения покрытия. На основе анализа этой информации формируется соответствующее предупреждение. Некоторые типы активных датчиков имеют также отдельную обогреваемую поверхность, которая позволяет определять возможность таяния и испарения льда и снега при повышении температуры.

Таблица 2

Сравнительная характеристика дорожных датчиков

Вид датчика	Измеряемые параметры	Достоинства	Недостатки
Пассивный	- Температура поверхности дороги; - Толщина водяной пленки до 4 мм; - Состояние дороги (сухо /влажно / мокро /лед);	- Простая конструкция; - Более низкая стоимость по сравнению с другими типами датчиков.	- Не прогнозирует опасность обледенения; - Находится в покрытии и подвергается механическому воздействию транспорта; - При наличии снежного наката датчик неинформативен
Активный	То же	- Более информативный по сравнению с пассивным датчиком; - Позволяет получать предупреждение об опасности обледенения; - Дает предупреждение при любых типах антигололедных реагентов	- Более дорогостоящий; - Находится в покрытии и подвергается механическому воздействию транспорта; - При наличии снежного наката датчик неинформативен
Бесконтактный	- Толщина водяной пленки; - Температура поверхности; - Наличие и толщина слоя снега на покрытии; - Сцепление/трение .	- Главное преимущество заключается в отсутствии необходимости устанавливать датчик в дорожное полотно; - Определяет наличие снега и толщину слоя; - «Обучается» различным погодным ситуациям; - Предоставляет информацию «сцепление/трение»	- Более низкая точность измерения температуры поверхности дороги (±0,8 °С); - Датчик является «интеллектуальным» и требует «обучения» по специальным алгоритмам, учитывающим особенности климатических условий; -Самый дорогостоящий
Мобильные	- Состояние поверхности дороги; - Температура поверхности дороги; - Сцепление (трение).	- Ведёт термокартирование в реальном времени; - Позволяет оптимизировать маршруты спецтехники и расход противогололедных реагентов	- Более низкая точность измерения температуры поверхности дороги (±0,8 °С); - Непостоянная работа на дороге; - Более дорогостоящий.

Основная сложность в эксплуатации таких датчиков состоит в том, что они заложены в дорожное покрытие и воспринимают механические воздействия от проходящего транспорта.

Такого недостатка лишены дистанционные интелектуальные датчики. Существуют микроволновые и инфракрасные дистанционные датчики, которые устанавливаются на опорах над поверхностью дороги и производят бесконтактное измерение температуры дорожного покрытия и оценивают состояние покрытия – наличие влаги, льда и снега [5].

Мобильные датчики могут быть добавлены к стационарной метеорологической сети. Они устанавливаются на транспортном средстве, на расстоянии 1-2 метра от поверхности дороги. Датчик определяет такие параметры, как состояние поверхности дороги, температуру полотна дороги, температуру воздуха, относительную влажность, температуру точки росы. Эти измерения дополняют информацию АДМС и снимают данные о дорожном покрытии между местами установки АДМС. Они позволяют сформировать «температурную карту» дороги, выявить наиболее «холодные» участки, где раньше начинается процесс обледенения.

Анализ таблицы 2 показывает, что определить снежный накат можно только с помощью бесконтактного датчика. Эти датчики имеют очень много достоинств по сравнению с другими типами датчиков.

Так как на дорогах Центральной части Российской Федерации наиболее часто образуется скользкость в виде снежных отложений, то все типы датчиков, кроме бесконтактного будут неинформативны. При наличии слоя снега на датчиках, заложенных в покрытие, они не дают показаний. Таким образом, для систем погодного мониторинга в России можно рекомендовать бесконтактные датчики.

Библиографический список

1. Васильев, А.П. Эксплуатация автомобильных дорог: В 2 т. Т.2: [Текст] : учеб. пособие для вузов / А.П. Васильев. – Москва. : Академия, 2010. – 319 с.
2. ОДМ. Руководство по борьбе с зимней скользкостью на автомобильных дорогах. - Утв. 16.06.2003. распор. Минтранса РФ № ОС-548-р. -М: Информавтодор, 2003. – 72 с.
3. Самодурова, Т.В. Погодный мониторинг в системе оперативного управления зимним содержанием автомобильных дорог [Текст] :обзорная информация/Т. В. Самодурова. – Москва: Информавтодор, 2006. –Вып. 2. - 45 с.
4. Самодурова, Т. В. Оперативное управление зимним содержанием дорог : Научные основы [Текст] /Т.В. Самодурова. – Воронеж : Изд-во ВГУ, 2003. – 168с.
5. Самодурова, Т. В. Метеорологическое обеспечение зимнего содержания автомобильных дорог [Текст] / Т.В. Самодурова. – Москва : ТИМР, 2003. –126 с.

СЕКЦИЯ 5. СЕЛЬСКОХОЗЯЙСТВЕННОЕ МАШИНОСТРОЕНИЕ: УПРАВЛЕНИЕ, РАЗВИТИЕ И ИННОВАЦИИ

УДК 622.6: 625.144.5

Пидуст Н.А. Актуальные направления развития современного оборудования для уплотнения грунтов

Current trends in the development of modern for compaction grounds

Пидуст Николай Александрович,
Студент 5 курса факультета механизации,
Новочеркасский инженерно-мелиоративный институт
им. А.К. Кортунова ФГБОУ ВО Донской ГАУ Новочеркасск, Россия
Pidust Nikolay Aleksandrovich,
5th year student of the faculty of mechanization,
Novocherkassk engineering institute reclamation A.K.Kortunova name FSBEE HE "Don State
Agrarian University" Novocherkassk, Russia

Аннотация. *Интенсивное расширение в области уплотнения земляного полотна ведет к необходимости развития и совершенствования техники уплотнения. В данной статье рассмотрены самые распространённые, способы укатки грунта, предложены возможные направления модернизации существующих дорожных катков.*

Ключевые слова: *классификация, дорожный каток, машина природообустройства, уплотнение дорожного полотна, валец, виброплита с вибровозбудителем.*

Abstract. *The intensive expansion in the field of compaction of the roadbed leads to the need to develop and improve sealing techniques. This article discusses the most common ways of soil formation, suggests possible directions for the modernization of existing road rollers.*

Keywords: *classification road roller, environmental machine, compaction of roadway, roll, vibroplate with exciter.*

Обеспечение мелиоративных и строительных мероприятий техническими средствами является одним из основных факторов интенсификации и повышения качества производимых работ. Большое значение в области строительства автомобильных дорог, дорожных оснований, аэродромов, а так же постройке насыпей и дамб, занимают дорожные катки, которые занимаются процессом уплотнения земляного полотна и подстилающих слоев а так же уплотнением асфальтобетонных смесей в дорожных покрытиях.

В уплотнении грунтов важную роль играют два условия:

— материал дорожного покрытия

— способ его укатки.

На данный момент укатка материала - это один из простейших и дешевых методов уплотнения, кроме того этот метод является широко распространенным в нашей стране. Необходимо понять что, для каждой местности и для каждого грунта необходим определенный тип катка. От работы именно этой машины по большей части зависит срок службы дорожного сооружения. [1]

Основным определяющим значением укатки материала является умение специалиста правильно определить необходимый для работы каток, используя следующие классифицирующие факторы:

— тип рабочего оборудования;

— способ укатки грунта;

По типу рабочего оборудования различают катки с гладкими вальцами, решетчатые, кулачковые, пневмоколесные, комбинированные.

Основной и наиболее простой вид катка имеет обечайку вальца с гладкой поверхностью, что подходит для уплотнения асфальтобетона и других ровных поверхностей по типу дорожного покрытия.

У катка с решетчатым вальцом, металлические стержни составляют решетчатое покрытие обечайки. Такое покрытие способно дробить твердые включения грунта, повышая тем самым качество уплотнения. Решетчатые вальцы можно использовать для уплотнения связных или несвязных видов грунтов.

На катке с кулачковым вальцом, кулачки закреплены в ряды, или в шахматном порядке вдоль всей обечайки вальца. Площадь контакта кулачков с грунтом создает напряжение большее в два, а то и в четыре раза больше, нежели напряжение под самой обечайкой. При уплотнении рыхлого грунта, особенно при первых проходах – кулачки целиком помещаются в него, в следствии в контакт входит валец катка. За счет уплотнения грунта уменьшается площадь контакта кулачков с поверхностью земли. Эффективность этого вида катков можно рассматривать только при уплотнении связных рыхлых грунтов, в которых толщина уплотняемого слоя варьируется в пределах 25 – 40 см. [2]

Необходимость повышения длительности прилагаемой нагрузки на уплотняемый грунт привела к созданию пневмоколесных катков. Проходя по уплотняемому материалу, пневматическое колесо, деформируясь на площади контакта с поверхностью, на десятые доли секунды создает повышенное напряжение необходимое для необратимой деформации грунта.

Нагрузка на одно колесо такого катка составляет около 5 тонн и за время напряжения успевает углубиться в земное полотно и дорожное покрытие на 30 см. Однако использование данных катков, по очевидным причинам, не возможно на рассыпчатых поверхностях, вроде песка.

Для расширения возможностей оборудования, современные катки выпускают комбинированного типа. Они снабжены рабочим оборудованием различного типа. Из них наиболее актуальны катки с вибрационными вальцами, а так же дорожные катки с пневмоколесами. Данные катки способны обеспечить наиболее обширную область применения. Универсальность этих катков позволяет работать с уплотнением таких грунтов как: асфальтобетонные смеси и суглинки, пески и грунты с крупными вкраплениями. [4]

Комбинированные катки снабжены шинами высокого давления, переняв их у пневмоколесных катков. Это позволяет уплотнять несколько слоев различного материала: специальные шины уплотняют верхний слой материала, а вибровалец создает давление на глубине под рабочей зоной шин. При строительстве покрытий необходима ровная поверхность материала, что и обеспечивает в этом случае валец с гладкой поверхностью.

По способу укатки грунта дорожные катки можно разделить на:

— статические;

— вибрационные.

В статическом дорожном катке принцип действия напрямую зависит от силы тяжести, грунт утрамбовывается под весом рабочего органа.

В вибрационном катке задействованы периодические колебания одного или нескольких рабочих органов, что в совместности с силой тяжести позволяет использовать меньшую массу РО. Воздействие на уплотняемый материал вибрации позволяет в 3-5 раз уменьшить число проходов по одному следу, в частных случаях виброкаток позволяет увеличить глубину уплотнения на 2м и более. Использование вибрационного принципа действия так же позволяет работать с крупнообломочным типом материала. [5]

Необходимо отметить что, не смотря на положительные качества, у подобного типа машин существуют и недостатки. Одним из основных является низкая эффективность вибрационного воздействия на материал при использовании круговых колебаний неопределенной частоты.

Способом решения этой проблемы может стать повышение производительности катка и снижение энергоемкости процесса уплотнения за счет использования тиксотропных свойств асфальтобетонных смесей и направленного воздействия вибрации на поверхность уплотняемого материала.

Поставленная задача может быть решена установкой на каток виброплиты, установленной перед ведущим вальцом, на тяговую раму, соединяющую виброплиту с рамой катка. На виброплиту мы предлагаем установить вибровозбудитель направленного действия для создания колебаний с частотой от 70 до 80 Гц и механизм регулирования угла возмущающей

силы в пределах 120°-135° к поверхности уплотняемого слоя в направлении движения катка, в зависимости от типа уплотняемой асфальтобетонной смеси. [4]

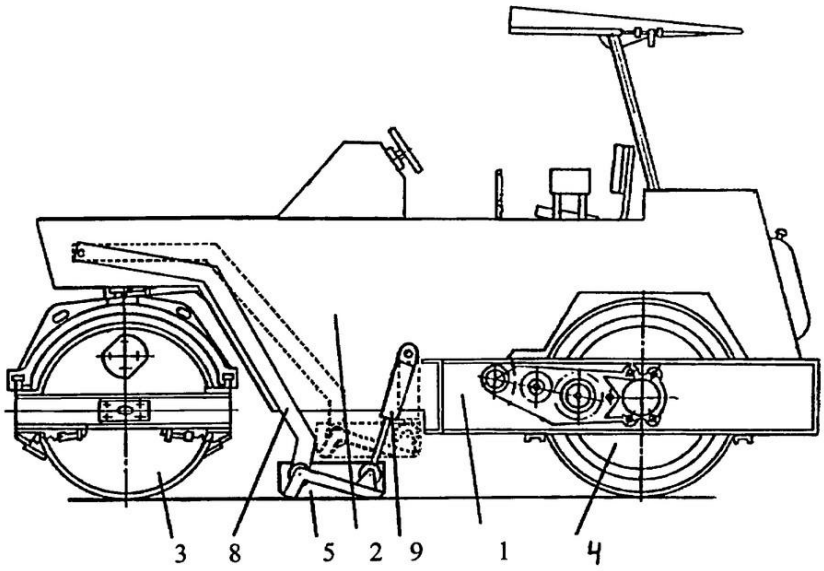

Рисунок 1. Общий вид усовершенствованного катка

Конструкция предлагаемого дорожного катка (рисунок 1) включает в себя раму (1), на которой установлена силовая установка (2), ведущий (3) и ведомый (4) вальцы. Виброплита (5), с установленным на ней вибровозбудителем направленного действия (6) и механизмом (7) регулирования направления действия возмущающей силы в зависимости от типа уплотняемой смеси, соединена с рамой катка через тяговую раму (8), гидроцилиндры (9) служат для подъема и опускания виброплиты. Двигатель привода вибровозбудителя (10) крепится кронштейном (11) к установочной плите (12).

При работе предлагаемого катка в начале уплотнения, когда в слое происходит интенсивное накопление остаточных деформаций, каток совершает 2-3 прохода с плитой, находящейся в транспортном положении. Затем после увеличения сопротивления смеси уплотнению и снижения интенсивности образования остаточных деформаций в уплотняемом материале, виброплита устанавливается на поверхность слоя гидроцилиндрами, обеспечивается необходимое статическое давление (0,01...0,03 мПа) и при начале передвижении катка включается вибровозбудитель. [3]

Использование виброплиты с высокой частотой колебаний (до 80...90 Гц) и направленной возмущающей силой позволит достичь определенных технических результатов:

— уменьшит сопротивление смеси уплотнению;

— изменит уплотняемый слой посредством тиксотропных изменений;

— повысит качество поверхности уплотняемого материала;

— снизит количество проходов по одному следу за счет активного накопления остаточных деформаций;

— повысит производительность катка и снизит энергоемкость процесса уплотнения.

Использование вышеописанных дорожных катков позволит значительно повысить производительность катков и максимизировать качество процесса уплотнения за счет использования тиксотропных свойств дорожного покрытия и направленного воздействия вибрации на поверхность уплотняемого материала.

Библиографический список

1. Технологические машины и комплексы в дорожном строительстве (произв. и техн. эксплуатация): Учеб. пособие / В.Б. Пермяков, В. И. Иванов, С.В. Мельник / М.:ИД «БАСЕТ» , 2014. С. 70-75.
2. Дорожные машины: В 2-х частях. Ч2.Машины для устройсства дорожных покрытий. Учебник для вузов по специальности «Строительные и дорожные машины и оборудование» /К.А. Артемьев, Т.В. Алексеева /Переиздание/ М.: ИД «КООМ» , 2016. С. 130-140.
3. Машины для земляных работ: учебник для студентов вузов/Доценко А.И. Карасев Г.Н. Кустарев Г.В./ М.:Изд. Дом «БАСЕТ» , 2014. С. 95-110.
4. Машины для земляных работ: Учебник/ Гаркави Н.Г, Аринченков В.И Карпов В.В./ Переиздание/ М.: ИД «КООМ» , 2017. С. 183-198.
5. Шестопалов К.К. /Строительные и дорожные машины: учебник для студентов высших учебных заведений// М.:ИЦ «Академия», 2015. С. 79-87.

Электронное научное издание

Наука, техника и технологии: глобальные и современные тенденции

сборник научных трудов по материалам I Международной научно-практической конференции

23 мая 2018 г.

По вопросам и замечаниям к изданию, а также предложениям к сотрудничеству обращаться по электронной почте mail@scipro.ru

Подготовлено с авторских оригиналов

ISBN 978-1-387-85809-5

Формат 60x84/16. Усл. печ. Л 3.5. Тираж 100 экз.
Lulu Press, Inc. 627 Davis Drive Suite 300
Morrisville, NC 27560
Издательство НОО Профессиональная наука
Нижний Новгород, ул. Ломоносова 9, офис 309