Block chain Technology LLC

The Districts project white paper

Last update: 11/10/2017

v 1.0.2

www.districts.io

Abstract. Nowadays, a growing part of funds' transactions and services is achieved through a peer to peer (P2P) Cash system known as Bitcoin; an open source crypto-currency developed by Satoshi Nakamoto team in 2008 and launched in 2009. The system relies on public-key cryptography and blockchain technology, granting an unprecedented level of anonymity. Exploiting this technology to a further extent, comes Districts, a 3D virtual reality platform promoting the crypto-currency to people with no computer science expertise, allowing them to build or use decentralized applications in a living world, for a better presentation_and management of their services, businesses, without writing or compiling code. In parallel, and based on the Bitcoin and Dash source code, with a characterized blockchain that enables many functions in Districts, we present to you 3DCoin, our new crypto-currency.

Table of content:

1. Introduction

Virtual reality and Blockchain should be seen as a new delivery presentation channel, a channel through which users can deploy applications in a secure space. Not only does this capability to address varying presentation styles' interaction, it also offers significant business advantages in cost, time and the ability to reach broadly dispersed audiences efficiently and effectively. Districts is a space where the creative possibilities of decentralized applications of shops, services, and management are unlimited; every user is free to opt for a project of interest to him. It is a self-regulating platform based on crypto-currency and ensures more attractive, impressive and efficient presentations for user's projects. It is an opportunity for small business owners who cannot afford the expenditures of advertisement and have difficulties in attracting customers from the neighboring or far communities. The platform assists their activities by enabling them to better present their shops, services, and business. Our mission is to help new users to be part of the crypto-currency community by promoting the existing blockchain based project using 3D virtual reality technology.

2. Districts:

Bringing everyone in a single continuous city ,with a total freedom of action, and displayed in high quality graphics yet very well optimized, Districts will be the place where your dreams come true, with all kinds of decentralized application running alongside, people will be immersed in a living world where cars, houses, shops, playfields, or any imaginable activity are of their own making, the user remains totally merged with reality, as services can be provided both in and out of Districts, thus giving a maximal exposure for the lowest cost to businesses.

It is designed as an assembly of **platforms** in the shape of concentric rings atop of water, the center holds all the starter facilities providing tutorials and examples of applications, next to them, the first ring serves as a showcase featuring inaugural pre-made buildings which are going to kick start the Districts' life, it is divided into specialized districts, gaming, business, and private. The subsequent growth involves a gradual creation of platforms triggered by the saturation of old ones.

Movement options are multiple both in standard and VR mode, targeted fluid relocation for short or long displacements, movements through devices like hover boards, or vehicles of all sorts, which can be customized. Unlimited movement on foot for VR is not possible with the actual technology yet, its' implementation will be done in the future.

The realism brought by the VR must also be followed by realism in **communication**; this is why vocal proximity chat is implemented, but traditional modes are also available, such as Voice on IP for smartphone simulation, or encrypted text messaging and mailing.

For **business,** the immersion brought by virtual reality enhances every possible interaction: real estate developers, car, or even clothing dealerships; will be able to present their products and their possible customizations in a realistic lively manner. These are shown in the city space or even in instances accessible from street or mall stores.

Customer acquisition will be simpler and cheaper due to the constantly available and renewed circulation incentives, which are induced to them by the presence of interesting activities in every corner of Districts. People are going to be constantly transiting; this is due to the great freedom allowing the community to express all its creativity.

The ability to remotely visit a house or a motel room and inspect every corner of it, try furniture in a virtual model of your house and see what fits it best, view and experiment a car in varying environments, will improve customers decision making leading to a better satisfaction.

Outside of Districts, creators efforts are heavily taxed in varied ways, in Districts, none of that. All revenues will go fully to their rightful owners.

Gaming in VR shatters every former experience, as it gives players the chance to live the action instead of simply and distantly triggering functions. But things are slowly taking off because game development requires complex skills, especially coding. In Districts, we believe that the possibility for everyone to bring their ideas to life, by the use of the **Districts' visual Studio**, will induce a boom in the field.

Education: Online learning shows a high dropping out rates, less than five percent of enrolled students finish the curriculums, and the obvious reason is that usual methods fail to maintain students' motivation on the subject, they get bored and become disengaged, virtual classes solve this problem, imagine a physics class where students are transported in front of a besieged city to learn about ballistics through the use of a catapult, or a coding class giving each student the opportunity to code his own personalized robot and see him getting more and more lively as the coding skills of the learner grow. Learning will just stop being boring when it is done in epic sceneries and with all kind of interactive setups.

Mental health can greatly profit from VR, because feeling the presence of a caring person has a better effect that any other online interaction like text dialogue or even video conferencing. Virtual anonymous gatherings for psychological support, one on one therapy, personalized phobia treatment with customized environment helping patient to gradually confront their fears, live ASMR sessions where the asmrtist can move around their audience and give them the most intense and realistic attention, these are a few of the many possibilities that can emerge.

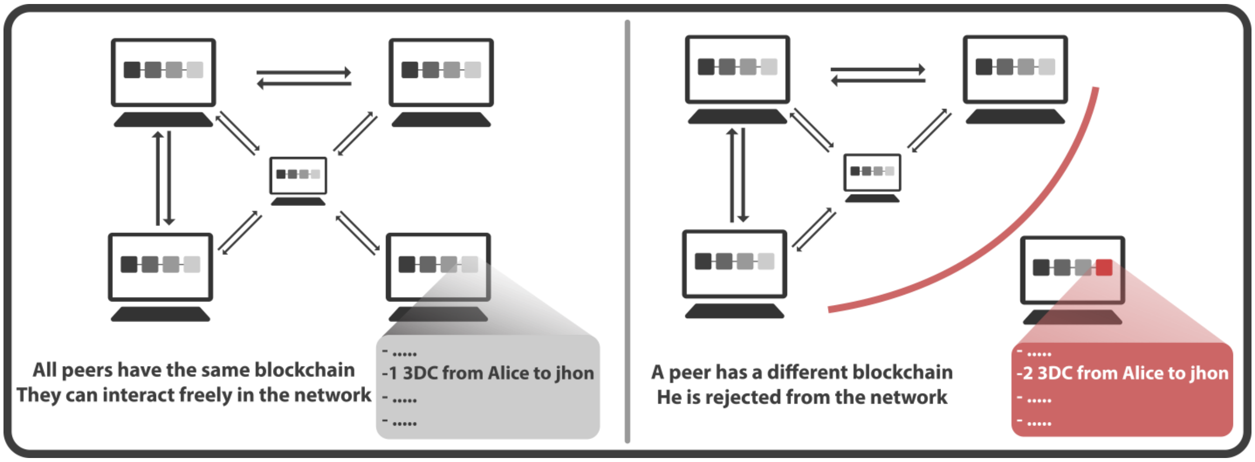
Private housing offers for users the possibility to create a personal customizable space that could have many purposes such as friend gathering spot, watch to movie together, play arcade or card games, it could even be a display gallery. The host must be online in order for other to access the private space.

Decentralization being a key element in our project, there is no central authority, meaning, that all content control is in the hands of the community, through a voting system, all data, except user specific data, will be stored in the **3DCoin blockchain** which doesn't only serve for the crypto-currency.

3. The 3DCoin blockchain

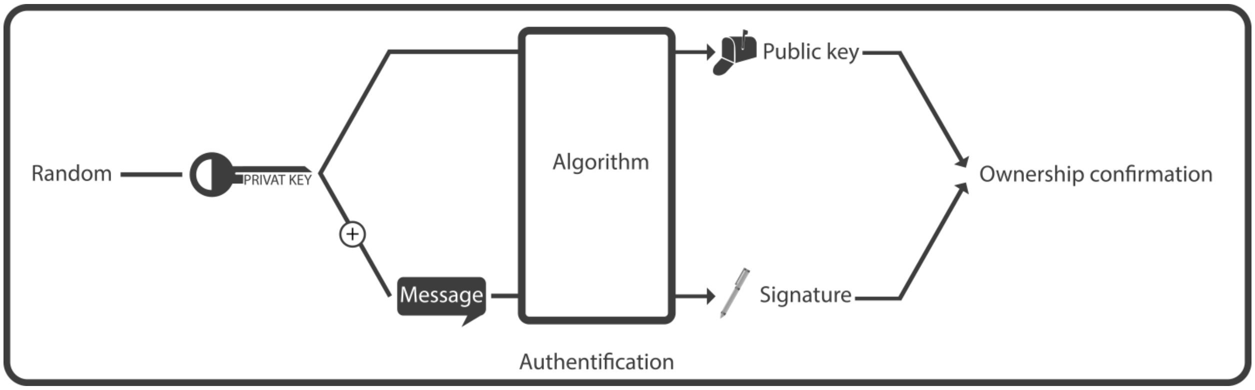
3DCoin is the new cryptocurrency that we have developed; the protocol was created using Bitcoin and Dash source code as a basis to inherit their safety and power, with implementations making it fully compatible with Districts, besides new features.

The blockchain is a system where storage and authentication of data and operations are performed in a conjoint and transparent way by every computer running the protocol, giving all users a total awareness and collective control over the process, rendering forgery impossible. Updates cannot be performed without consensus, unlike usual systems, it does not rely on a central entity, as mathematical functions do all the work, these functions or any aspect of the protocol are publicly available because of its open-source nature. For currency, the blockchain can be considered as a public ledger, for applications, it can be considered as a decentralized server. [1]

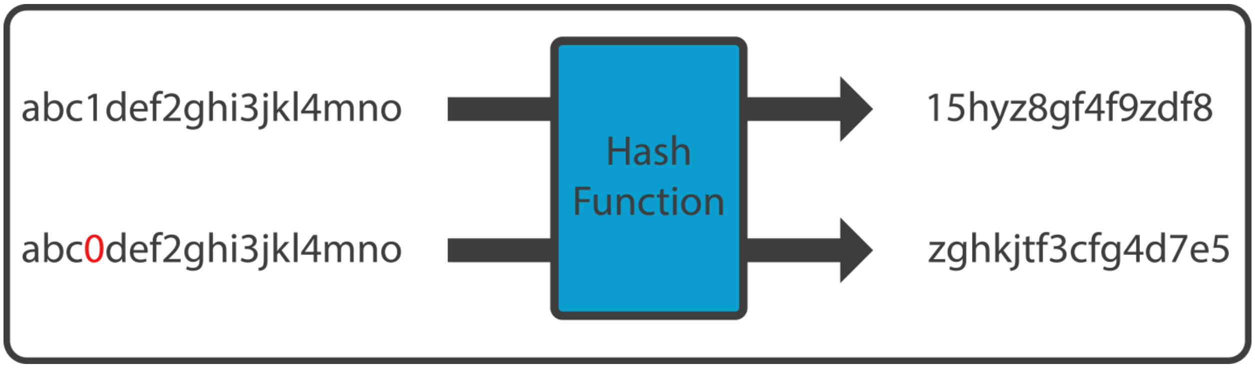

All peers have the same blockchain
They can interact freely in the network
- .....
-1 3DC from Alice to jhon
- .....
- .....
A peer has a different blockchain
He is rejected from the network
- .....
-2 3DC from Alice to jhon
- .....
- .....

Identity is based on public-key cryptography, it uses a pair of keys, a **Private** one that must be kept secret by the user, linked through an elliptic curve function to a **Public**-key, which can be used as an address to receive money. When the owner wants to spend his 3DCions, he broadcasts a request message with a signature attached to it; the network then can verify that the message comes from the owner to confirm the transaction.

The 3DCoin system enables the use of **many** public-key cryptography methods, improving safety, and allowing the use of preexisting keys from other crypto-currencies, or any other system, for example, if you use a PGP key for your emails, you will also be able to use it to receive 3DCoins.



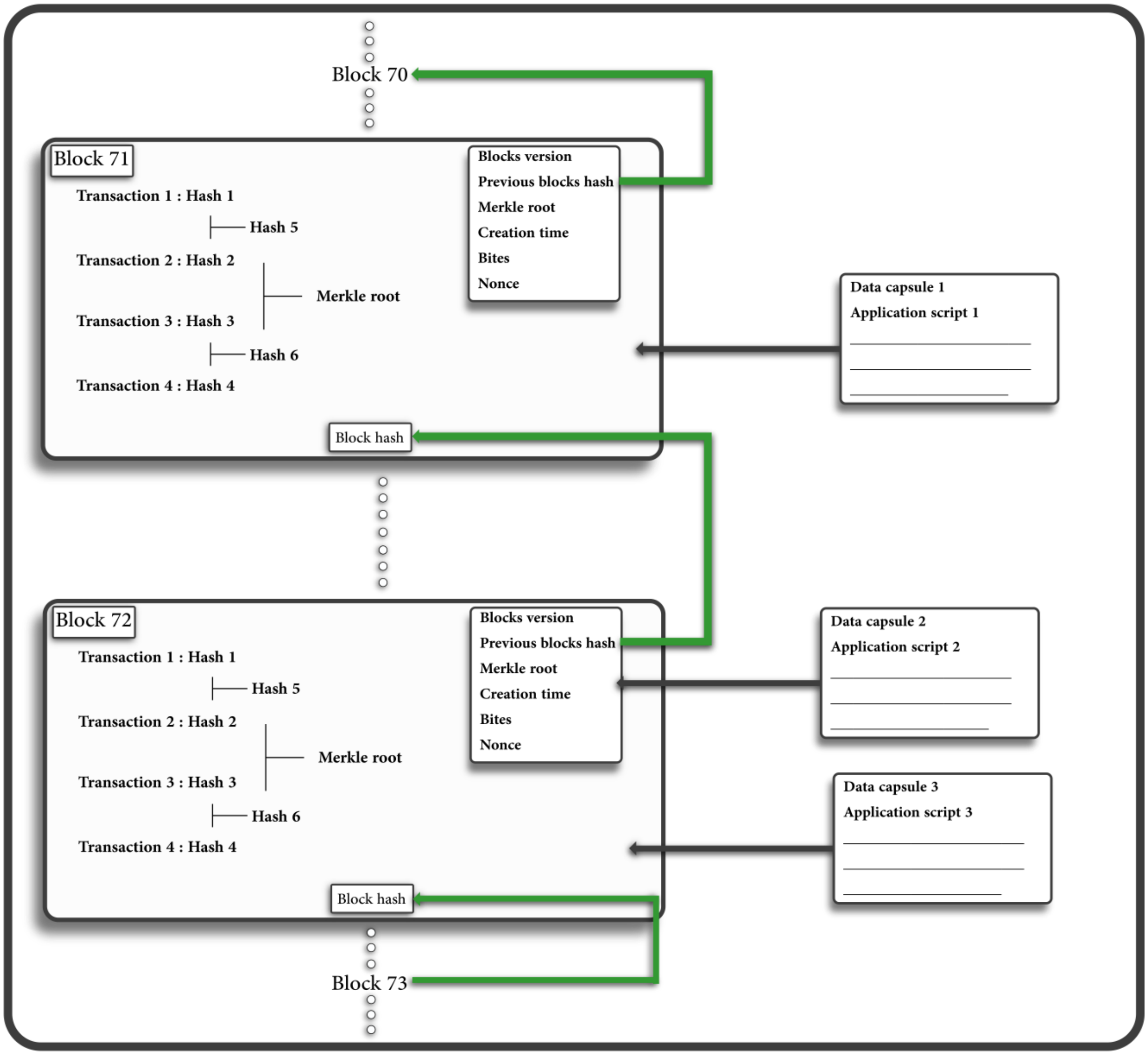
A **Hash** is a series of characters generated by a one way function from another series, an input gives a unique output that change drastically at the slightest change in the input, this is why hashing is the backbone of authentication. [2]



A **Merkle tree** is a fast and secure method used to verify the content of large data structures; it is made of a branch sequence of hashes that ends with a single hash, the **Merkle root**, in the blockchain, the merkle tree is based on transactions.

Blocks: Data of all kind, be it transactions history or buildings references, are recorded in the blockchain, which is made of 8 megabits blocks that are linked to form a simple chain, where every block refers to the hash of the previous one, in each, transactions are saved in a particular order forming the base of a Merkle tree, of the 8 megabits, 1 will be dedicated to feeless small transactions.

To each block, a number of **Data Capsules** can be attached; these hold the **scripts** of **Decentralized applications**, their size has a maximum of one megabits.

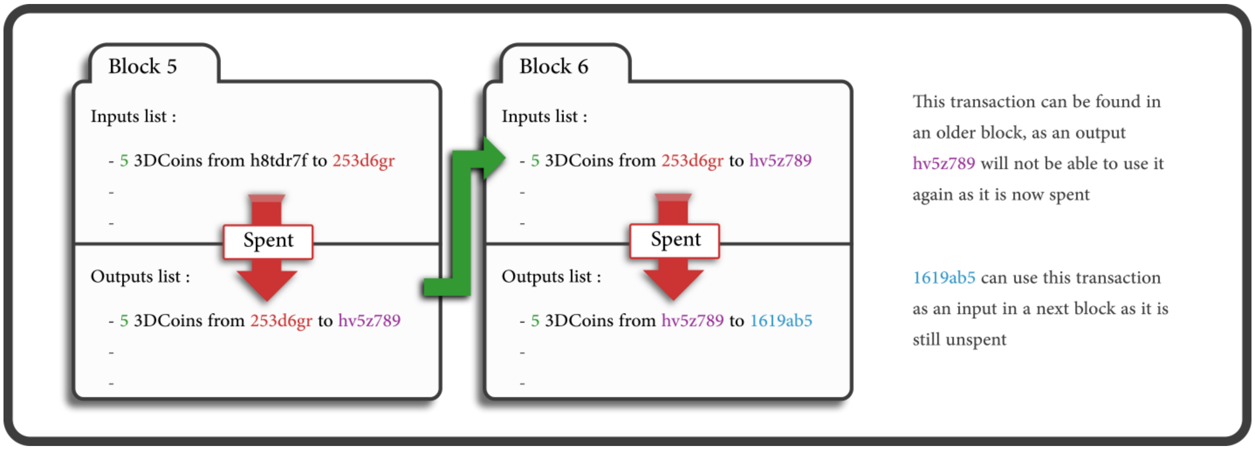


Transactions and balance:

Transactions are the inalterable units from which the balance is calculated, they hold the amount of transferred 3DCoins, the public-key of the sender and the receiver. A transaction comprises a micro fee paid by the sender that goes to miners; it can grow with the size of the transaction, as transactions can have **scripts** attached to them, these are used to **program transactions**, or to define ownership of **land parcels** and **decentralized applications** in districts.

Unlike a normal ledger, balance in the blockchain is not noted as a particular amount, it is defined by the inalterable transactions history saved periodically on the blocks. To his key, the user owns incoming transactions; the wallet application will browse all the blockchain looking for the unspent transactions and calculate the balance from them.

When the user wants to send money, the **wallet application** will broadcast a message containing the incoming transactions he wants to use, an output transaction directed toward the public-key he wants to transfer money to. Verifications of both ownership and sufficiency of inputs are performed by peers, until the final addition in a block, at the base of the merkle tree.


Block 5
Inputs list :
- 5 3DCoins from h8tdr7f to 253d6gr
-
-
Spent
Outputs list :
- 5 3DCoins from 253d6gr to hv5z789
-
-
Block 6
Inputs list :
- 5 3DCoins from 253d6gr to hv5z789
-
-
Spent
Outputs list :
- 5 3DCoins from hv5z789 to 1619ab5
-
-
This transaction can be found in an older block, as an output
hv5z789 will not be able to use it again as it is now spent
1619ab5 can use this transaction as an input in a next block as it is still unspent

Transactions that define ownership of **land** and **decentralized applications** have no 3DCoin amount in them; they refer the data Capsule that holds the application script, the owner, and the parcel coordinates.

Scripts:

3DCoin uses a scripting system for transactions; the Script is **stack-based**, and processed from **left** to **right**. It is Forth-like, purposefully not Turing-complete, with no loops.

A script is essentially a list of instructions recorded with each transaction that describe how the next person wanting to spend the 3DCoin being transferred can gain access to them. The script for a typical 3DCoin transfer to destination 3DCoin address D (an address is the hash of a public key) simply encumbers future spending of the 3DCoin with two things: the spender must provide:

- A public key that, when hashed, yields destination address D embedded in the script.
- A signature to show evidence of the private key corresponding to the public key just provided.

Scripting provides the flexibility to change the parameters of what's needed to spend transferred 3DCoins. For example, the scripting system could be used to require two private keys, or a combination of several, or even no keys at all. [3]

A **stack** is a data structure based on the principle of <LIFO>: *last in first out*, meaning that the last added element is the first to be removed. Stored data behaves like a pile, with two main operations, **Push**: adds an element on the top of the pile, **Pop**: removes the last element added to the pile.

3DCoins script specificities

To improve the programing possibilities, many standard operations where enhanced into taking arguments, the direct effects are a better script readability, and a massive decrease in the complexity of code planning, instead of having a limited set operations that interact with specific stack items only, the programmer will be able to employ the stack in all its depth with extremely short commands. The OP prefix was removed because the limitation it was set to get around became irrelevant (function names cannot begin with a number), there are no more multiple variants of a function, variants are unified into one function that accept arguments previously pushed in the stack.

For example:

Bitcoin has three duplication operations (OP_1DUP, OP_2DUP, OP_3DUP), the first duplicates to top stack item, the second duplicates the top two stack items, and the third duplicates three.

The duplication is now done with a single function, DUP (without op), to duplicate an item, a number <x> must be pushed before the execution of DUP, <x> represent the place in the stack. If the argument is made of many numbers separated by commas <x,y,z>, it means that multiple items are duplicated, the DUP operation will read the item, **pop** it, and will do its work according to the argument, if the argument is made of two numbers separated by a dash, it will be considered as a range, <x-y>, all the items from x to y are duplicated. The same improvement is brought to many other functions.

Smart script optimization is automatically performed by reducing unnecessary steps that might be made, resulting in shorter but equally functional scripts, saving allot of space in blocks and data capsules.

Script verification module adds a safety layer by checking all possible operation sequences, and argument, allowing only bug free and always spendable transactions, and decentralized application to be accepted in the network, this is done by all peers.

The **script** is written with specific **words** representing functions, and is executed from left to right [4], a basic script as an example**:**

⟶<numberA><numberB> <1,2> EQUAL, from left to right:

- <numberA> pushes the numberA in the stack
- <numberB> pushes the numberB in the stack
- <1,2>Argument for the EQUAL function is pushed in the stack.
- EQUAL pops the argument, checks the equality between the two top stack items (<1,2>Top and Second stack items) and pops them, pushes 1 in the stack if correct, and 0 if not.

Here is the script for a **standard transaction**; it is a combination of two scripts:

scriptSig (<sig> <pubKey>), found in the **output** transaction. **scriptPubKey**, found in the input transaction (<1> DUP <160>HASH <pubKeyHash> <1,2><EQUALVERIFY <1,2>CHECKSIG).

Combined:

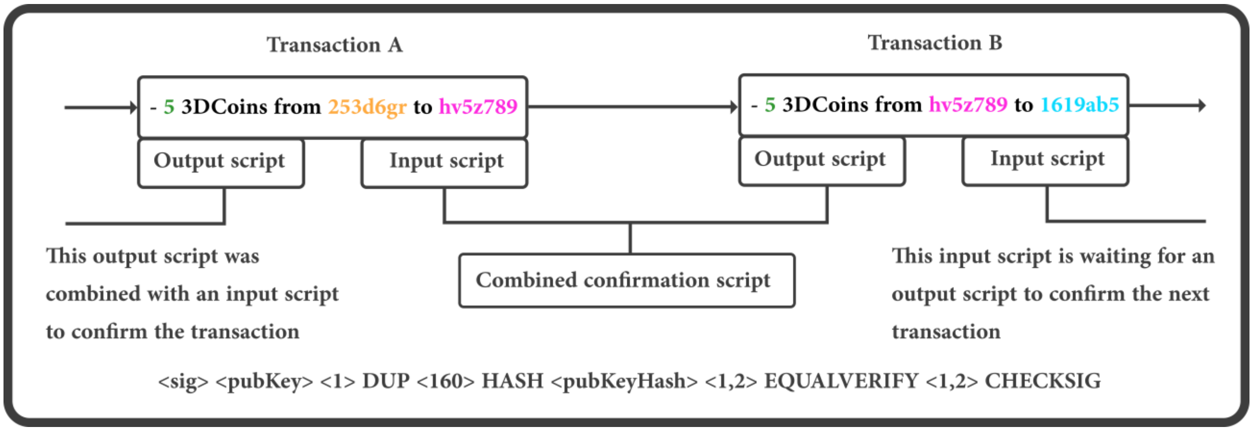
⟶ <sig> <pubKey> <1>DUP <160>HASH <pubKeyHash> <1,2>EQUALVERIFY <1,2> CHECKSIG

This script begins first by verifying that the public key displayed by the spender is the same as the public key of the input, and then verifies the signature. **(Reminder**: Operations always pop the arguments)

1- <sig> pushes the signature to the stack; <pubKey> pushes the public key to the stack.
2- <1> Argument for DUP operation.
3- DUP duplicates the top stack item (pubKey), and then pushes the copy in the stack.
4- <160> Argument for HASH operation.
5- HASH, the top stack item (pubKey) is popped, hashed, then the hash in pushed in the stack.
6- <pubKeyHash> pushes the inputs' public key hash to the stack.
7- <1,2> Argument for EQUALVERIFY operation.
8- EQUALVERIFY pops the argument, checks equality between the two top stack items then pops them, pushes True in the stack if equals, then pops it, invalidated the transactions if unequal.
9- <1,2> Argument for CHECKSIG operation
10- CHECKSIG, Checks the signature with the two top stack items (sig and pubKey), then pops them, returns True when correct, meaning that the transaction is confirmed, invalidates the transaction if not.

Operations	**Stack**
	Empty
<sig><pubKey><1>	<sig><pubKey><1>
DUP	<sig><pubKey><pubKey>
<160>	<sig><pubKey><pubKey><160>
HASH	<sig><pubKey><pubHashA>
<pubKeyHash>	<sig><pubKey><pubHashA><pubKeyHash>
<1,2>	<sig><pubKey><pubHashA><pubKeyHash><1,2>
EQUALVERIFY	<sig><pubKey>
<1,2>	<sig><pubKey><1,2>
CHECKSIG	TRUE

To summarize: All transactions contain both an **output** script, and an **input** script, every time a user wants to spend **an existing transaction,** he must deliver a correct **output** script that when combined with the **input** script of the input, true is returned, he must also provide a new **input** script in the **newly created output** transaction to define the future spending conditions. Scripts are generated automatically by the wallet application, but they can also be manually written.



Transaction A
- 5 3DCoins from 253d6gr to hv5z789
Output script
Input script
Transaction B
- 5 3DCoins from hv5z789 to 1619ab5
Output script
Input script
This output script was combined with an input script to confirm the transaction
Combined confirmation script
This input script is waiting for an output script to confirm the next transaction
<sig> <pubKey> <1> DUP <160> HASH <pubKeyHash> <1,2> EQUALVERIFY <1,2> CHECKSIG

Nodes

These are peers that must hold an entire copy of the blockchain, their job is to collect, verify, and assemble transactions in a particular order to prepare the creation of blocks, but only one block can be added to the chain at a time, this is why the **Mining** mechanism is used.

Mining is about delivering a proof of work; the goal is to lower the probability of the creation of two blocks at the same time, and it is done by running a set of block unique data through The **NIST5** hashing algorithm**,** which is a combination of the 5 finalists in the NIST hash function competition (BLAKE, Grøstl, JH, Keccak, Skein), when the resulting series of numbers meets a certain criteria, the block is considered to be mined, and is broadcasted to the network to be added to the blockchain after taking the said series of numbers as its **hash**. (Mining can be done with GPU-CPU, any hardware)

Once a block is mined, a **block reward** that consists of newly generated 3DCoins is distributes among **Miners (50%)**, **Super** n**ode operators (45%)**, and a **Development reserve (5%)**. This is the unique 3DCoin generation method.

In very rare occasions, more than one block are mined at the same time, presenting a situation where peers have a different last block added to their blockchain, in this case, they simply synchronize to the majority best chain.

When the computing power increases in the network (**hash rate**), either by the growth of the miners' number or their hardware performance, a compensation must be made by raising the mining difficulty, thus keeping a healthy block production rate, and maintaining the consensus over a single chain.

Super nodes

The owner of a **node** can upgrade it to a **Super node** by staking **1.000** 3DCoins that he can recover at any time; this allows him to contribute to diverse functions, and be **rewarded** for that, with such a stake, Super node **operators** are incentivized to make the best decisions possible, as they will directly benefit from the effects, the functions are:

- Casting his vote for content control.
- Participating in the Instant transaction service.
- Hosting Coin blending sessions.

Development Reserve

5% of the **block reward** will be allocated to a reserve fund; it will pay for projects that benefit 3DCoin at the end of each month. Funding from the reserve will be used to pay developers and other contributors, as well as integrations with major exchanges and API providers.

The super node operator receives one vote, which he can cast in favor of the proposal he wants to support, proposals and votes will be held in a proper site.

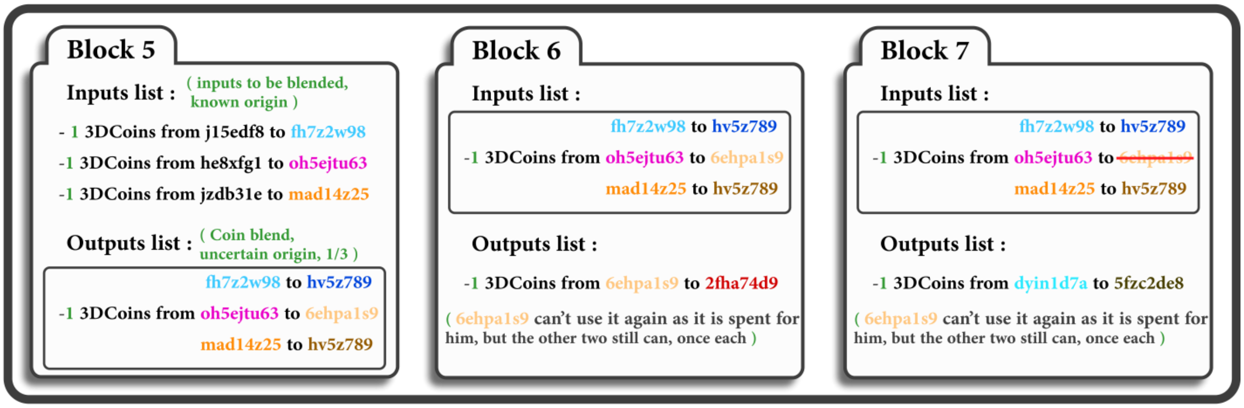
Coin Blend

Some uses of the public-key disclose the identity of the owner, like a physical order where the key and the delivery address can be linked, or a key placement in a forum signature for donations. To regain the anonymity of funds, the Coin blending service is offered.

The Blending request starts from the wallet application, where the user selects the transaction he wants to anonymize, the wallet will break it down into standard denominations (10, 1, 0.1, and 0.01) to facilitate the operation which will be managed by a super node; then checks the list of present super nodes and randomly selects one of them using the hash of a block, for a maximal randomness.

The transaction is placed in a Blending queue, when three matches for a standard denomination are found; the super node casts a readiness alert to the three wallet applications, the Blending session starts at their confirmation.

Each of the three wallets sends both the input to be mixed, and a **newly generated** public key that will receive the anonymized funds, from them, the super node creates a triple transaction containing the three inputs, and the three receiving public keys. The triple transaction is sent to the three wallets, they verify the exactitude of the inputs and the public-keys, sign the transaction, and send it back to the super node. At this point, the super node has a triple transaction with three signatures, after a last verification, this transaction is broadcasted like any other one to be added in a mined block, and finally into the blockchain.


Block 5
Inputs list :
(inputs to be blended, known origin)
- 1 3DCoins from j15edf8 to fh7z2w98
-1 3DCoins from he8xfg1 to oh5ejtu63
-1 3DCoins from jzdb31e to mad14z25
Outputs list :
(Coin blend, uncertain origin, 1/3)
fh7z2w98 to hv5z789
-1 3DCoins from oh5ejtu63 to 6ehpa1s9
mad14z25 to hv5z789
Block 6
Inputs list :
fh7z2w98 to hv5z789
-1 3DCoins from oh5ejtu63 to 6ehpa1s9
mad14z25 to hv5z789
Outputs list :
-1 3DCoins from 6ehpa1s9 to 2fha74d9
(6ehpa1s9 can't use it again as it is spent for him, but the other two still can, once each)
Block 7
Inputs list :
fh7z2w98 to hv5z789
-1 3DCoins from oh5ejtu63 to ~~6ehpa1s9~~
mad14z25 to hv5z789
Outputs list :
-1 3DCoins from dyin1d7a to 5fzc2de8
(6ehpa1s9 can't use it again as it is spent for him, but the other two still can, once each)

The result: a transaction containing three equal inputs, that can be spent once by each of the three receiving public-key. The receiving public-keys being **newly created**, no one can know to whom they belong, as neither or the three inputs refer to a particular key.

Instant transactions

Even if transactions take a few minutes to be confirmed in the blockchain, it isn't fast enough for retail environment, where goods are meant to be delivered instantly, this is where the instant transaction comes at hand:

The process is orchestrated by 10 randomly selected super nodes; these are changed every time a block is mined. When an instant send request is sent, the super nodes call the entire network to lock the input that will be treated; any transaction using it will be rejected. Once the entire network is synchronized on the lock; the super nodes will verify and then confirms the instant transaction even if it is not saved in a mined block yet, it will eventually be broadcasted like any other transaction and treated as such.

But one can ask the following question: why aren't all transactions instantaneous if it is so safe? It is totally possible, the option to make all your transactions instantaneous by default is available, but, these transactions require more work from the network, thus costing higher fees, also, instantaneity is not necessary for all operations, a few minutes confirmation time are not a big deal when ordering an item that will be delivered after two or three days.

Decentralized Hosting

All Applications scripts are encoded and stored in the **Data Capsules**, removing the need for a central server, or at least, making it optional. The Districts space will be divided among nodes, as they do not only manage messages and blocks, they also host the actions and events happening in the Districts, bandwidth saving will be done by a loading priority based on range, actions will not be loaded beyond a certain distance. For games, a peer to peer method combining the decentralized and distributed models has been developed, instead of communicating directly with each other; peers use nodes as bridges to gain a better routing and faster synchronization.

Sub-Currencies

Non fungible personalized assets can be created with the wallet application, this creation consists on a **reversible** conversion from 3DCoin, only the **name** must be chosen by the user as the **rate** is always 100 sub-currencies to 1 3DCoin, sub-Currencies can be used as tokens, shares, coupons or in any possible way, for example, 100 **SC** could be created to represent the stock of a company (100%), and distributed among shareholder, then freely traded.

In the blockchain, sub-currencies are simply 3DCoins scripted to be perceived as different by the **Wallet application.** [5]

External address support

Many cryptography methods are offered to users including the ones of other crypto-currencies; it means that our network can accept any public-key from any other crypto-currency thus, simplifying multiple account management. You can receive 3DCoins to your old key, with no need for the creation of a new one. Our long term goal is the generalization of this use among all crypto-currencies, allowing people to pick a universal key, it'll only happen when most crypto-currencies will agree on one shared signature verification algorithm.

Transactions can be simple transfer orders, but our protocol allocates an important portion to **scripts** in blocks' space, making the implementation of function such as **conditioned transactions** possible, beside standard transactions.

4. Conditional transaction

Transactions can be programmed to be received only if certain conditions are satisfied, this offers a large spectrum of options in terms of automated operations. The said conditions are written in the **script** that is **attached** to the transaction.

A **simple visual tool** will be available in the wallet application to set such transactions; the resulting broadcasted message contains both the transaction and the conditions script. The main conditional parameters are **Time**, **Public-key**, **and Password**. A few examples of conditioned transactions:

1- Abstract example:

Bob sends 3DCoins to **Alice**, **Alice** can use them only after 24 **hours**, at a specific store **Public-key**, using a specific **Password**, and if not spent before 48 **hours**; the money goes back to **Bob**.

2- Delivery setup example:

Alice sends **password protected** 3DCoins to **Bob, Bob** Sends a product to **Alice; Bob** will have access to the 3DCoins only when **Alice** enters the **Password** in the **3DCoin mobile app** running on the delivery man's smart phone. The script:

> **Output script**: <sig><pubKey><password>
> **Input script**: <160>HASH <SavedPassHash> <1,2> EQUALVERIFY <1>DUP <160>HASH <pubKeyHash> <1,2>EQUALVERIFY <1,2>CHECKSIG

In the stack:

Operation	**Stack**
	Empty
<sig><pubKey><password>	<sig><pubKey><password>
<160>HASH	<sig><pubKey> passHash>
<SavedPassHash>	<sig><pubKey><passHash><SavedPassHash>
<1,2>EQUALVERIFY	<sig><pubKey>
<1>DUP	<sig><pubKey><pubKey>
<160>HASH	<sig><pubKey><pubHashA>
<pubKeyHash>	<sig><pubKey><pubHashA><pubKeyHash>
<1,2>EQUALVERIFY	<sig><pubKey>
<1,2>CHECKSIG	TRUE

3- **Secured escrow setup example**: (automatically generated by the wallet app)

Bob sends 3DCoins to an **escrow** (**John**), **John** can only send them to **Alices' Public-key** or return them to **Bobs' Public-key**, but can't use them himself, if not sent to **Alice** in a **Month**; the money goes back to **Bob**.

Some function definitions:

DUP: Takes the top stack item as an argument, pops it, then duplicates the new top stack item.

CUT: Execute **DUP**, and drop the original item.

IF: Checks the top stack item, if Zero, pops it and executes the ELSE operations, if not Zero, Executes the IF operations.

HASH: Takes the top stack item as an argument, pops it, reads the new top stack item, hashes then pops it, and pushes the hash in the stack.

EQUALVERIFY: Checks equality between the two top stack items, if correct, the script keeps going, if false, the script is ended and the transaction is rejected, pops both items.

CHECKTIME: If the top stack item value is less than the current Unix-time, TRUE is pushed in the stack, if higher, FALSE is pushed instead.

Constant definitions:

<Time>: Pushes the Unix Time deadline defined by Bob in the stack.

<pubKey>: Public key of the person trying to spend the transaction.

<pubKeyTo>: Public key of the person that is mean to receive the 3DCoins.

<PubKeyHashA,B,J>: Stored addresses of Alice, Bob, John.

Output script: <pubKeyTo><sig><pubKey>
Input script: <Time>CHECKTIME IF <1>DUP <160>HASH <PubKeyHash**B**> <1,2>EQUALVERIFY ELSE <1>DUP <160>HASH <PubKeyHash**J**> <1,2> EQUALVERIFY ENDIF <3>CUT <160>HASH <PubKeyHash**A**> <PubKeyHash**B**> 3,1|2 EQUALVERIFY <1,2>CHECKSIG

The script begins by verifying the deadline and the person trying to spend the transaction.

If conditions are satisfied, the receiver of 3DCoins can either spend the transaction directly, or redeem it; in this case, he will receive from the conditional transaction, a **standard** transaction totally free of fees.

When a conditional transaction is sent, there is always a default return time that can be changed, the goal is to avoid the cases of trapped 3DCoins when conditions are not met by the receiver.

5. Decentralized applications

Dapps are ones that use the blockchain technology. The visual side is held in the unreal engine library that will be part of the Districts' client, but the script, which is generated by the **Districts Visual Studio**, lies in the blockchain, in a **Data capsule**, cutting hosting costs and getting rid of downtime, for example: When a 3D model is used, its position and other properties, which are very light on memory, will go in the blockchain, but the model file itself or the texture are loaded from the user's PC where the client is installed. Dapps can range from games to educational platforms, or stores and showrooms, selling **real** (clothes, houses, food**)** or **virtual** products (programs, in game items, hosting services); there is literally no limit on what dapps can be.

The ownership of the Dapp is defined in a scripted transaction, one that doesn't hold any 3DCoin amount, but refers the Data Capsules where the Dapps script is saved.

A Dapp can be freely or conditionally used, but only the owner can modify its script. Scripts will be encoded to have a very small size, while being client specific, meaning that they can run only with Districts' client, ensuring a maximal safety, as the possible functions that a script can use are well defined and generated only by the **DVS**.

Any modification or update will be done through an update scripted transaction, broadcasted then verified by the network (identity, and safe script verification). If a script holds non-recognized functions, the update is rejected.

To attain our goal of making App creation accessible to everyone, **Districts Visual Script** was developed.

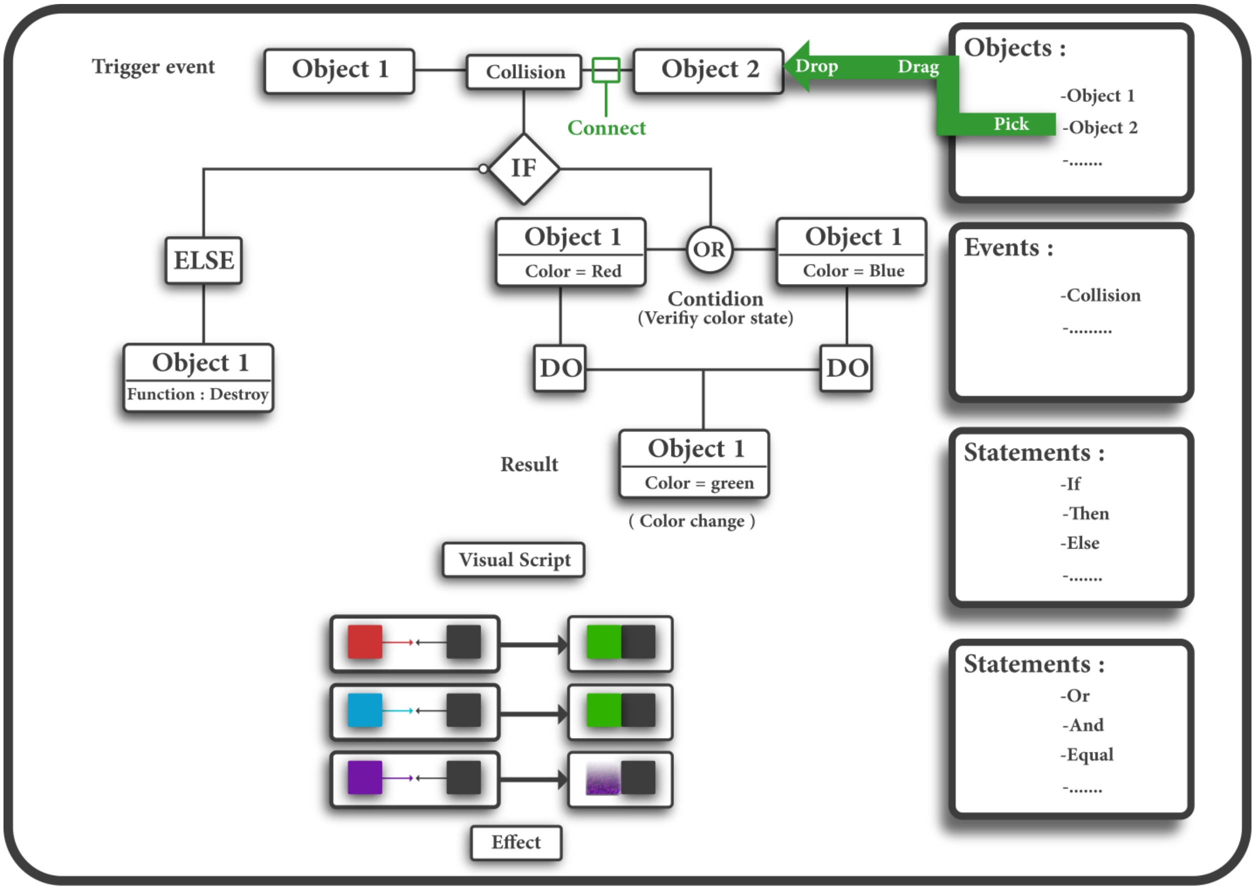
6. Districts visual Studio

The DVS enables anyone to create any program without coding; it presents a rich and expandable library of functions, textures, sounds, and 3D models, with an intuitive interface simplifying the creation of Applications. Users have a virtual reality edition mode that gives them the ability to build their Dapp in its final form, also, ready to use customizable templates are available for many Dapp types (examples of dapps types).

Tutorials that teach every aspect of DVS development can be found in the starting area of Districts in the form of interactive bots, as well as videos, or a dedicated forum section where community mutual aid will take place.

Pick drag drop and connect is the item placement method (3d models, buttons, lighting…), texture assignment, animation allocation, or even **visual scripting**, as most of the creation will be done by the **mouse** or VR controllers. Terrain modeling has a specific tool allowing the creation of varied landscapes (terrain noise, relief, ramps, erosion). Item properties menu can be opened with a click. Interactions and triggers are set in the interactions map, where processes are created as trees of conditions and results, the user draws his map, and the **DVS** turns it into an application.

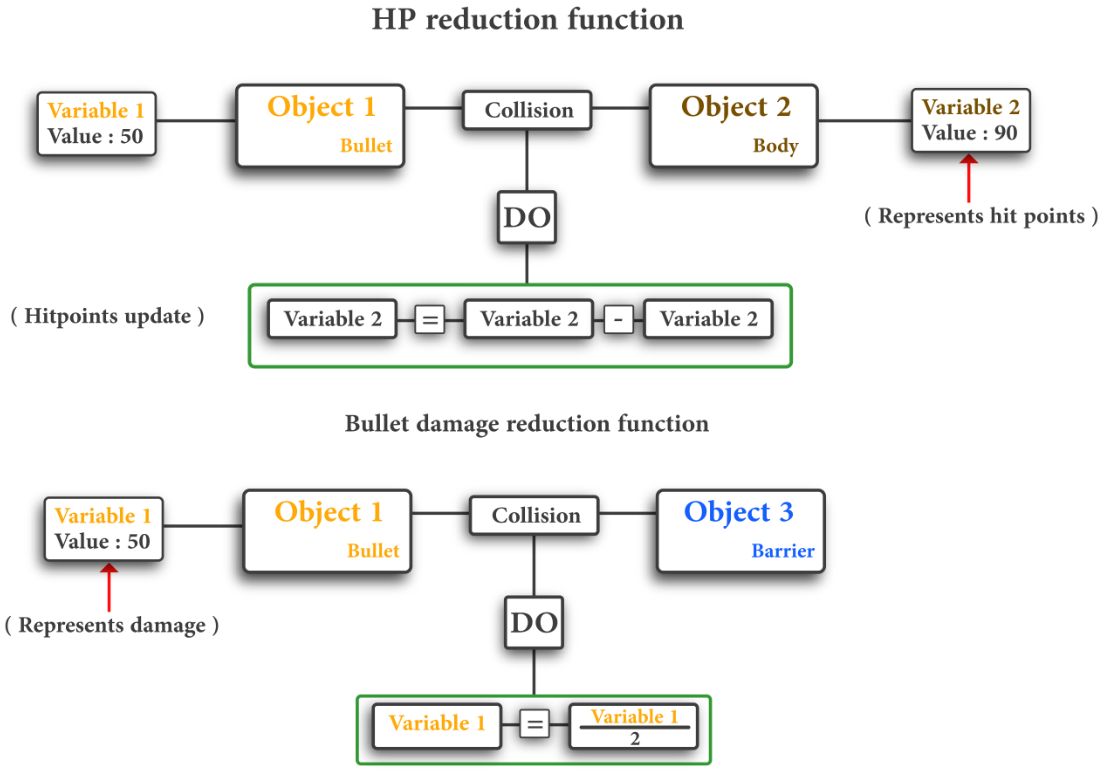
Visual scripting map example: Color change on collision



Body armor damage reduction

HP reduction function:

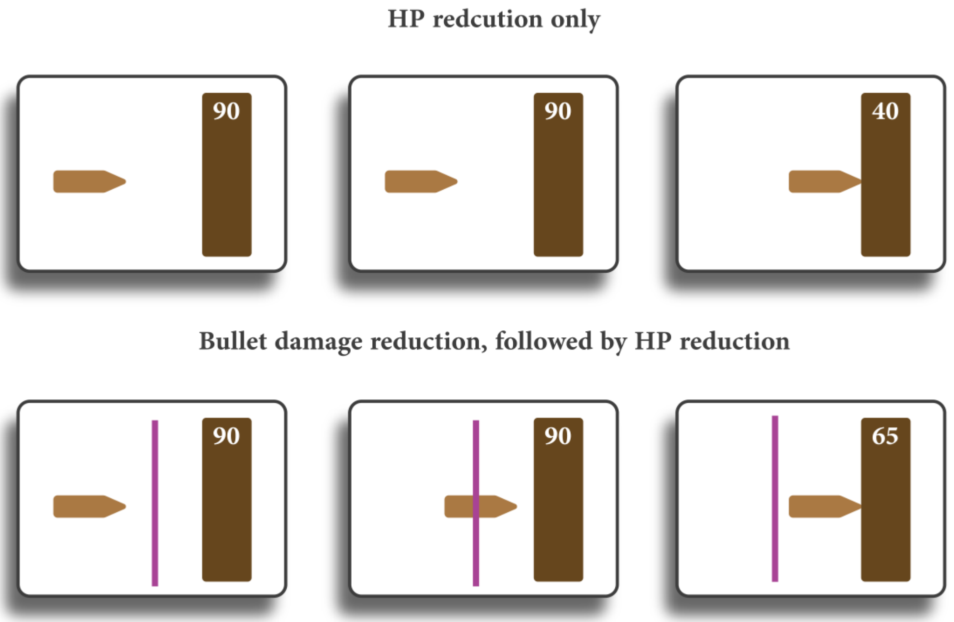
When the Bullet collides with the **Body**; a new value is given to the Hit points' value, by subtracting the **Damage** value from the actual **HP** value-Bullet



HP reduction function

Bullet damage reduction function

Visual Script



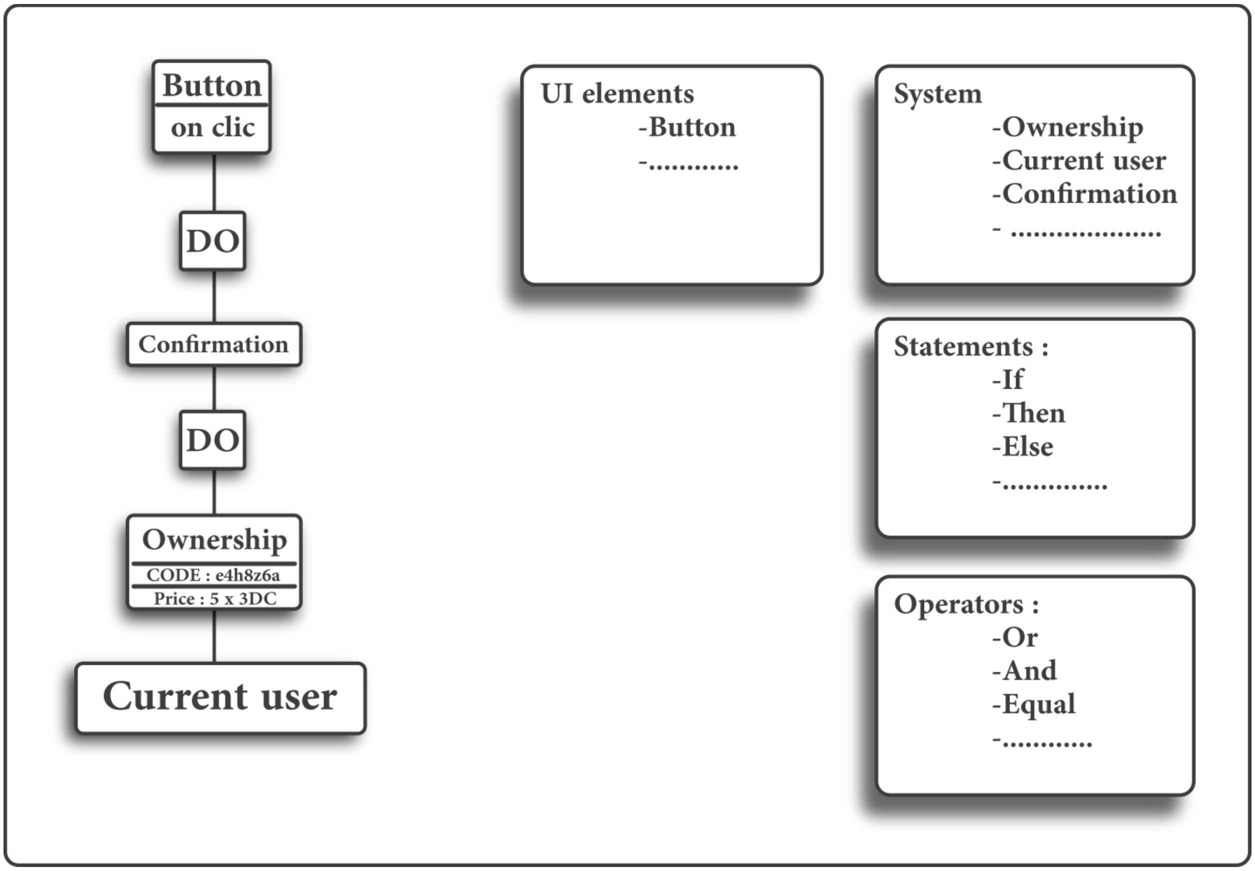
HP redcution only

Bullet damage reduction, followed by HP reduction

Effect

Payment function

In Dapps, ownership of sold products or access to services is defined by a scripted transaction, for example, when the right to use a service or a product is purchased, the **Ownership function** creates a payment transaction that contains a specific code, the Dapps check for the presence of such a transaction in the blockchain each time a person tries to use it. Creating a sale in a Dapp menu is as simple as any other function:



When the “**Purchase**” button is clicked, a confirmation is asked from the user, when he clicks on “Yes”, the **Ownership function** automatically creates and broadcasts an ownership transaction.

Before the release, the user can test his Dapp in an offline mode, once tested and checked; an **encoded script** is generated by the **DVS** and hosted in a **Data Capsules**, besides, if new models or textures are included, they will be submitted as additions to the **districts’ library**.

Criminal or inappropriate content can be reported by any user to be visually restricted after a **vote**. Only the people running **super nodes** can vote, they’ll make the most rational choice because of their direct investment in the system and its success. A portion of the block mining reward will be distributed among them as an **incentivisation** into participating in regulation.

Once the decentralized application is completed, a land parcel must be acquired to receive the building hosting it, defined by a **Land token.**

7. Land tokens

These define the ownership over a parcel that will be created to receive users' made or procedurally generated buildings or any structure, parcels are linked to decentralized apps which they host, both can be freely traded. Ownership of land will be set and saved in the same scripted transaction as the Dapp.

The user will have to select his parcel either in the pre-partitioned and pre-categorized Districts, or in a sandbox District where he can freely choose the size and spot.

The first 25 land tokens, that represent in the first ring of platforms, will belong to top investors, who can use them or trade them as they please; every other parcel is to be purchased with **3DCoins**, either from the system, in which case the 3DCoins will be added to the block reward, or from other users.

8. Conclusion

The districts project offers a complete framework where the highest expectations of consumers will be met by an incomparable offer, in terms of quality and quantity, creativity will be liberated, and ideas will never be confined in the realm of imagination, literally everyone will have the opportunity to express himself, and enjoy the creations of others, in a truly free market ruled by its users.

References:

[1] https://keepingstock.net/explaining-blockchain-how-proof-of-work-enables-trustless-consensus-2abed27f0845

[2] https://www.cryptocompare.com/coins/guides/how-does-a-hashing-algorithm-work/ (midified)

[3]https://en.bitcoin.it/wiki/Script

[4]https://github.com/BlockchainTechLLC/3dcoin/blob/master/src/script/interpreter.cpp#L688

[5] https://en.bitcoin.it/wiki/Colored_Coins