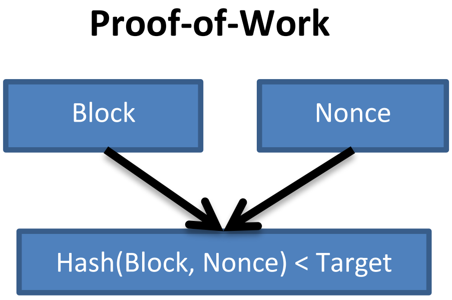
Motocoin

Motocoin is a crypto-currency of a new kind. It is inspired by Bitcoin but has major difference. Bitcoin is based on a scheme called *proof-of-work*. In this scheme miners perform some computations (called work) on their computers and then use proof of that work to secure coin transactions and to get their reward. Motocoin is similar to Bitcoin but instead of relying on a work done by computers it relies on a work done by humans. This work consists of playing 2D moto-simulator game, hence the currency name. People play the game and get rewarded for this. We call this scheme *proof-of-play*.

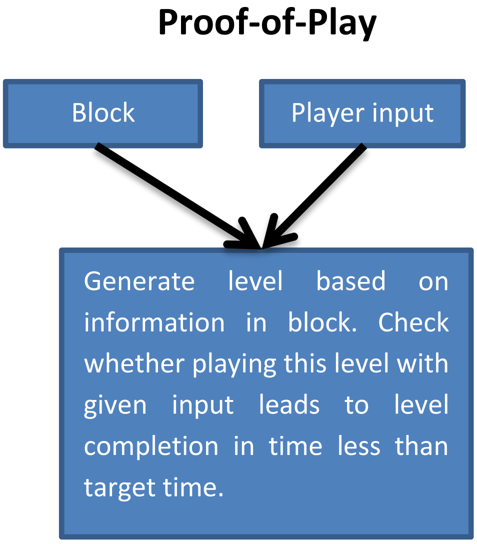


Proof-of-Work

Block

Nonce

Hash(Block, Nonce) < Target



Proof-of-Play

Block

Player input

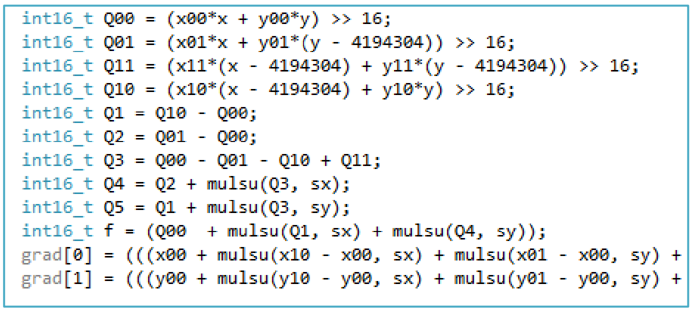
Generate level based on information in block. Check whether playing this level with given input leads to level completion in time less than target time.

Proof-of-work can be briefly described as follows. You have block and you try to find number (called nonce) so that hash of block together with nonce would be less than given target value. Once found it is easy for everyone to check that it is correct.

Proof-of-play is different. Instead of searching for number you are trying to find input to game such that level pseudo-randomly generated from block can be completed with this input. Instead of comparing hash with target value you are checking that time that it took to complete level with this input is less than given target time. Because having only one level per block would be bad there is additional nonce value (not shown on diagram above) that is used in level generation. Any node can check that given level can be completed with given input in required time.

Determinism

Game for proof-of-play should be deterministic. Physics engines are usually implemented using floating point arithmetic. But the problem with floating point numbers is that depending on CPU, compiler and compiler options they may give slightly different results. This is unacceptable for cryptocurrency because result must be the same for all peers, otherwise network will be forked. For this reason game in Motocoin is implemented using only integer arithmetic.



```
int16_t Q00 = (x00*x + y00*y) >> 16;
int16_t Q01 = (x01*x + y01*(y - 4194304)) >> 16;
int16_t Q11 = (x11*(x - 4194304) + y11*(y - 4194304)) >> 16;
int16_t Q10 = (x10*(x - 4194304) + y10*y) >> 16;
int16_t Q1 = Q10 - Q00;
int16_t Q2 = Q01 - Q00;
int16_t Q3 = Q00 - Q01 - Q10 + Q11;
int16_t Q4 = Q2 + mulsu(Q3, sx);
int16_t Q5 = Q1 + mulsu(Q3, sy);
int16_t f = (Q00  + mulsu(Q1, sx) + mulsu(Q4, sy));
grad[0] = (((x00 + mulsu(x10 - x00, sx) + mulsu(x01 - x00, sy) +
grad[1] = (((y00 + mulsu(y10 - y00, sx) + mulsu(y01 - y00, sy) +
```

Example of "integer magic" used in Motocoin source code.

Replay

Each replay is stored as list of player actions, e.g. press gas, wait 5 sec, release gas, rotate left, etc. It is possible to make only 60 actions in one play, this is necessary to keep the blockchain small. Without this restriction it would be possible to produce replays with thousands of actions and to severely bloat the blockchain. Each action is stored as 16-bit integer which includes delta time to previous action and action itself. If a is action then $\lfloor a/12 \rfloor$ is delta time to previous action and $(a \bmod 12)$ is action itself. Time is measured in ticks, there are 250 ticks in second. Each replay takes no more than 960 bits.

Difficulty adjustment

There is no problem to adjust difficulty in proof-of-work. Network hashrate can be computed based on mining speed and then it is easy to calculate necessary target value. But in proof-of-play there is no such thing as network hashrate, therefore other approach is necessary.

The following method for difficulty adjustment is used in Motocoin:

1. We know the time period that it took to mine last 1008 blocks, denote this time by T. Also we have a list of times (in-game times, not real-life intervals between blocks) with which levels were completed in last 1008 blocks.

2. Find median time in this list. This is such a time that half of the levels were completed faster than it and half slower than it. Denote this time by M.

3. We know that if target time was equal to M than last 1008 blocks would be mined no longer than in $2T$ time period.

4. Now we know how long it took to mine last 1008 blocks with current target time and we assume that if it would be equal to M than it would take twice as long. So we can use linear inter/extra-polation to find new target time with which last 1008 blocks would be mined with necessary speed (that is approximately 1 block in 5 minutes).

5. If new target time is less than M then use M instead. If new target time is greater than 60 seconds than use 60 seconds instead.

This algorithm is conservative that is it returns larger target time than it could be. With this algorithm target time will never become too small to make mining impossible because in last 1008 blocks at least 504 levels were completed in less time. It is yet remain to be seen how well it will approach target block generation speed of 1 block in 5 minutes, but at least it will never make mining impossible and it will decrease speed to prevent too fast block generation.

Bots

There are concerns that it is possible to make bots for this game. This is based on belied that “everything is botable”. But in many games (for example Go) humans are still superior to computers. It isn’t easy to make bot for game used in Motocoin. Pure brute force approach cannot be used because there are too many possible actions that player can do and search tree grows too fast. Some very smart algorithm is necessary that will require a lot of research. It is also necessary to distinguish levels which can be completed from the ones that cannot to not waste time trying to complete them, this is also not an easy task. Although theoretically bots are possible, I believe that this is not going to happen anytime soon. If that would actually happen, that would not be an issue for Motocoin, because the difficulty adjustment algorithm would prevent the bots from generating blocks too fast. It will be like transition from CPU -> GPU -> ASIC for Bitcoin.

Level generation

Levels are generated using Perlin noise. There is a good description of Perlin noise: http://webstaff.itn.liu.se/~stegu/TNM022-2005/perlinnoiselinks/perlin-noise-math-faq.html

Perlin noise is in fact a function $f(x, y)$. We assume that there is ground where value of this function is less than certain threshold and sky if it is greater.