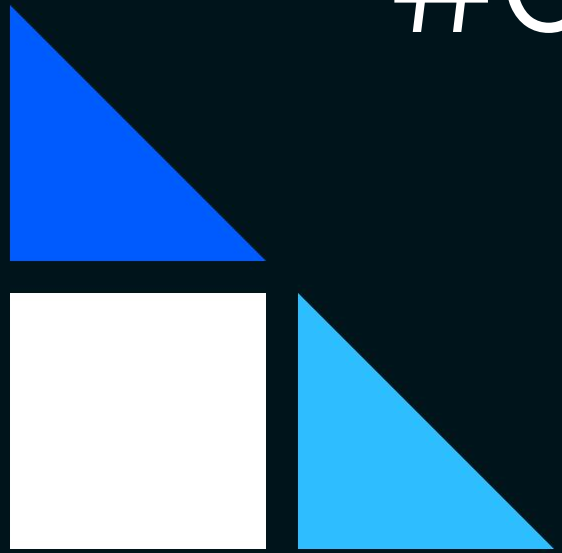
Blockchain In Rust

#04: Transactions 1



geeklaunch

not a geek? start today!

Before we start

Download and install Rust if you want to code along: https://www.rust-lang.org/

If you're on Windows, you'll also need Microsoft Visual C++ Build Tools 2017: https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2017. Alternatively, it's *very easy* to install Rust on the Windows Subsystem for Linux (WSL).

If you're on Linux, you'll need some form of GCC, which probably came preinstalled on your system.

Optionally, you may also want to install Git: https://git-scm.com/

The code written in this series can be found at: https://github.com/GeekLaunch/blockchain-rust

A Transaction

Alice has 50 coins.

Bob has 7 coins.

Alice sends Bob 12 coins.

Is that all?

Not quite.

Blockchain != spreadsheet

Transaction Verification Requirements

We have to protect against:

- Overspending (Where did the money come from?)
- Double-spending (Is the money available?)
- Impersonation (Who owns the money and who is sending it?)
- … (there are more, but we're just going to cover these three today)

List of rules for a Bitcoin transaction:
https://en.bitcoin.it/wiki/Protocol_rules#.22tx.22_messages

The Blockchain as a “Distributed Ledger”

What does it mean to “own a coin?”

Block 123	Block 124	Block 125	Block 126
Jaime ➔ Andrew (15) Chris ➔ Alice (50)	Francis ➔ Chris (34) Michiko ➔ Bob (7) Terrence ➔ Georgia (87)	**Alice ➔ Bob (12)** Zach ➔ Jaime (2) Chris ➔ Terrence (18)	Chris ➔ Zach (3) Zach ➔ Chris (2) Chris ➔ Zach (10) Zach ➔ Sophia (18)

Structure of a Transaction

Inputs

Outputs

Oh, and also…

Inputs are Outputs

Regular Transactions

For us right now, transactions only contain two important pieces of information:

- Set of inputs (which are *unused* outputs from previous transactions)
- Set of outputs (*new* outputs that can be used in future transactions)

From here we can calculate...

- ...the value of the transaction: $\sum inputs$
- ...the value of the fee: $\sum inputs - \sum outputs$

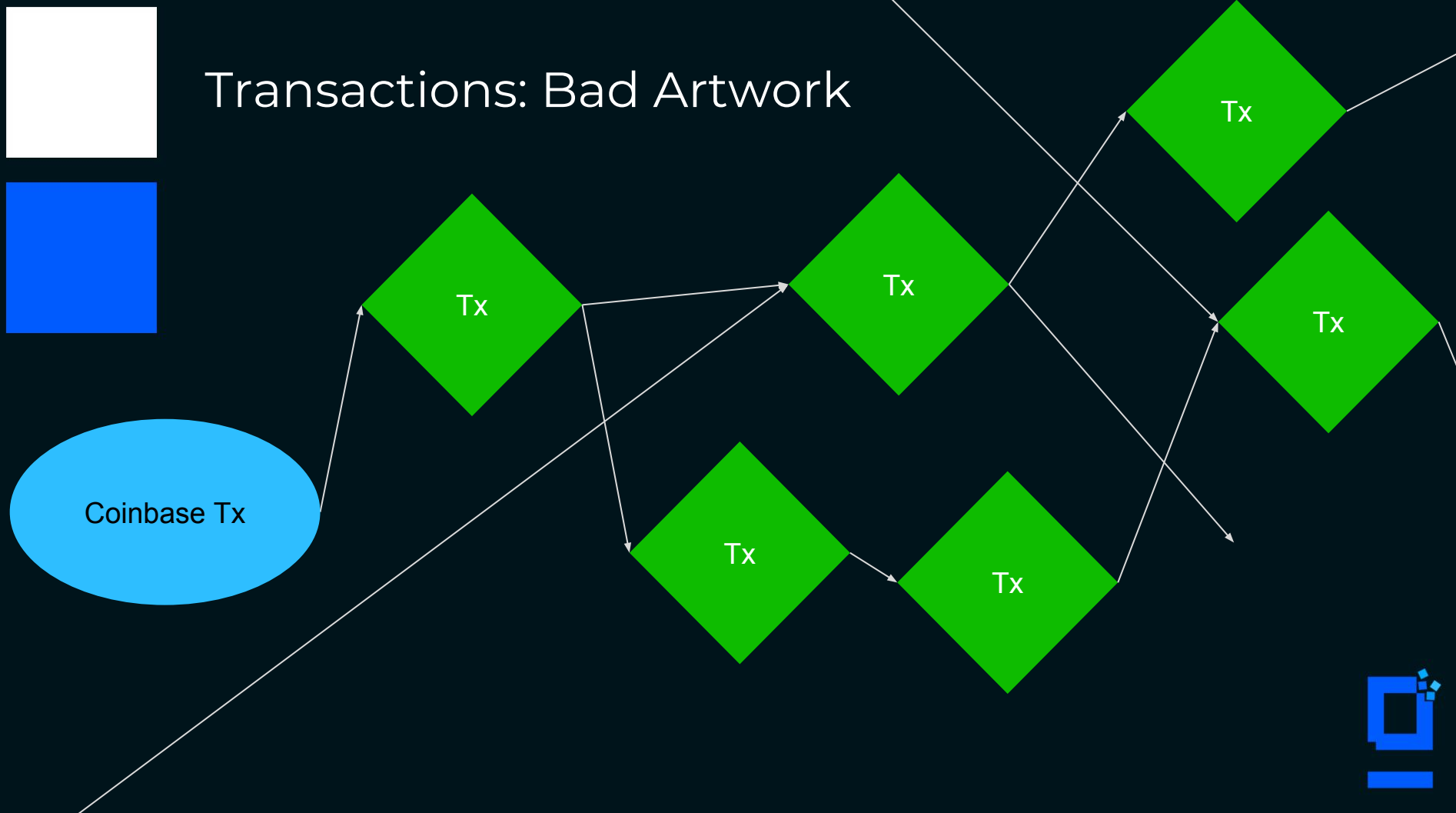
Coinbase Transactions

Where it all starts!

Coinbase transactions...

- ...do not require inputs
- ...produce an output
- ...allow the miner to collect all the transaction fees in that block and that block's block reward (coin genesis!)

Transactions: Bad Artwork



Recap

A transaction will take a set of outputs as inputs, and generate a set of outputs in turn.

Example:

[50] ➔ [12], [36]

50 ➔ 48

And does this mean that coins can get destroyed just like that? Nope! Miners take the leftovers (in this case 2) as their fee. If a transaction does not have room in it for a fee for the miner, what incentive does the miner have to add the transaction to their block?

Meeting Tx Verification Requirements

Overspending

Simple: the sum of the values of the inputs must be greater than or equal to the sum of the values of the generated outputs.

I can’t input 5 coins and be able to output 7.

Double-Spending

Make sure that any one output is never used as an input more than once.

This can be done by maintaining a pool of unspent outputs and rejecting any transaction that tries to spend outputs that don’t exist in the pool.

Impersonation

This can be solved by adding a cryptographic “signature” to outputs to verify they’re being spent by their owner.

We can’t assume that whoever *sent* us the transaction over the network is also the person who *created* the transaction.

For now, we’ll kind of ignore solving this problem. We might come back to it when we go over smart contracts.

Further Reading

Bitcoin Transaction Rules: https://en.bitcoin.it/wiki/Protocol_rules#.22tx.22_messages

Coinbase Transaction: https://bitcoin.org/en/glossary/coinbase

How is a transaction's output signed? https://bitcoin.stackexchange.com/q/45693

What is a double-spend? https://bitcoin.stackexchange.com/q/4974