

DATA BUDDIES SURVEY

**Help improve the student experience at
UW Madison**

Complete the survey by February 17th

Win one of TEN Amazon gift cards!




<https://tinyurl.com/data-buddies>

TOKENS & EXCHANGES

Kai Mast
CS639/839
Spring 2023

TODAY'S AGENDA

- Recap of last week's lecture
- More smart contracts
 - Interfaces
 - Tokens
- A break 
- Decentralized exchanges

RECAP: DECENTRALIZED APPLICATIONS

Definition: Application that is not controlled by a single, centralized entity

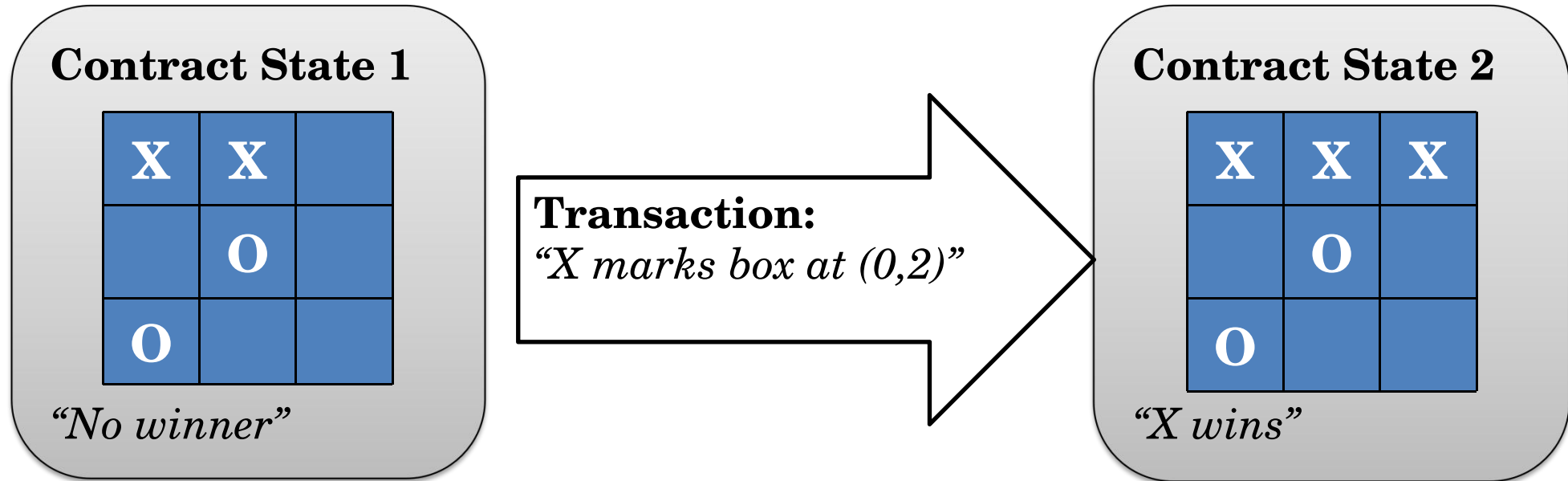
Why?

- Protect against a strong adversary, e.g., cheating administrators
- Make sure the application stays available

Can be built

- into a blockchain protocol, e.g., a cryptocurrency
- on top of a blockchain protocol, as a set of smart contracts

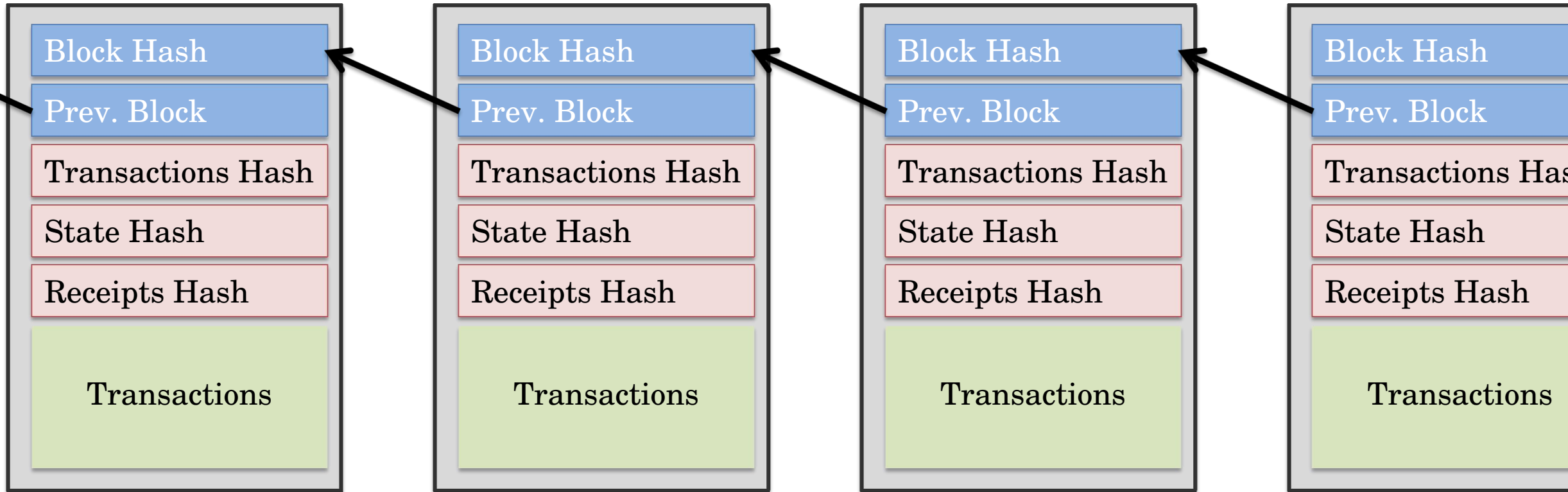
RECAP: SMART CONTRACTS



Two core feature:

1. Can encode arbitrary data (=state) onto the blockchain
2. Can execute arbitrary computation on that data (Turing completeness)

RECAP: THE ETHEREUM BLOCKCHAIN



Each block encodes a cryptographic digest of the state and the output/outcome of a transaction (receipt)

GAS IN ETHEREUM

Purpose of Gas: Pay for computation and transaction validation

Each transaction specifies

- **STARTGAS:** Initial gas budget
- **GASPRICE:** Cost per unit of gas (in Ethereum)

Maximum Transaction Fee: $\text{STARTGAS} * \text{GASPRICE}$

(Behavior before the London upgrade)

GAS IN ETHEREUM

Gas usage is calculated based, for example:

- **Base fee:** Cost of validation
- **Computation:** Sum of the cost of each computation step taken
- **Function calls:** Fixed cost for context switch
- **Stored Data:** Charge gas proportional to the size of the stored data
- **Deleted Data:** Get a refund proportional to the removed data

You can find the exact gas prices here:

<https://github.com/wolflo/evm-opcodes/blob/main/gas.md>

(Behavior before the London upgrade)

GAS IN ETHEREUM

What happens when a transaction runs out of gas?

- All changes of the transaction will be reverted
- Issuer of the transaction will still be charged for the execution

What happens when a transaction does not use its full budget?

- Issuer of the transaction will be refunded the different

CALLING OTHER CONTRACTS

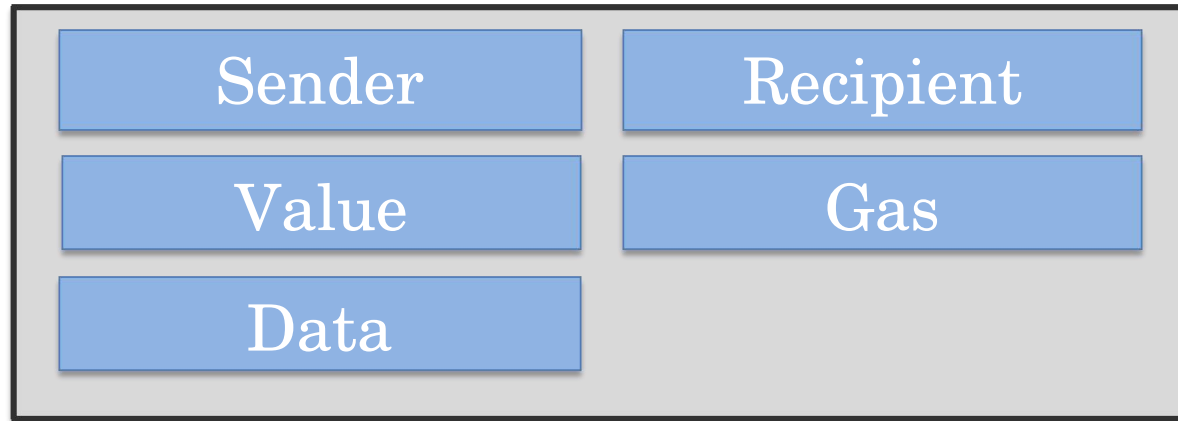
Goal: Build more complex example by combining multiple smart contracts

- Smart contracts can call other contract's functions during execution

Example: Exchanging Tokens

- Each token is managed by a dedicated smart contract
- Exchange is also a smart contract
 - Exchange calls both tokens to facilitate a trade

MESSAGES



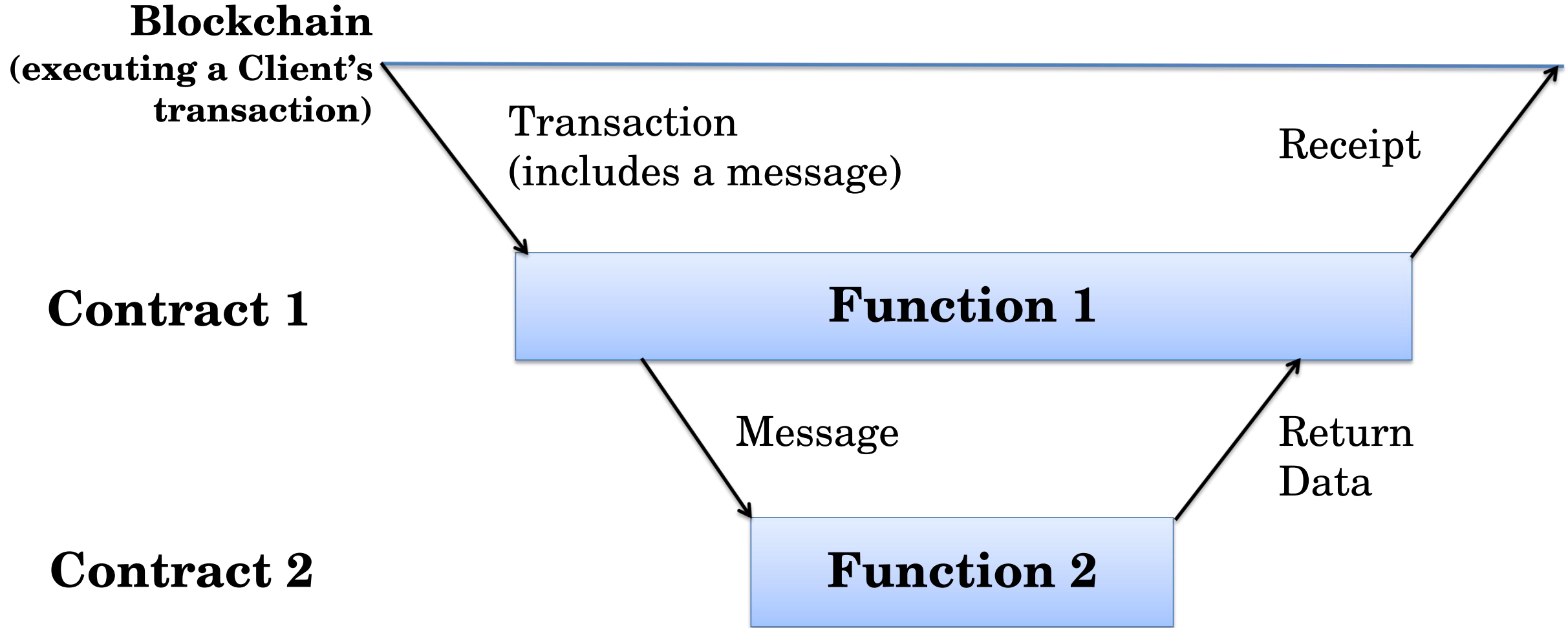
Messages contain information about a function call sent from

- one contract to another
- or a transaction to a contract

Messages between the contracts are not stored on the ledger

- Can be regenerated by executing the associated transaction

MESSAGES



TRANSACTION RECEIPTS

- Certifies the output of a transaction
- Like state, they are not stored on the blockchain
- Unlike state, they cannot be accessed by smart contracts

What do they contain?

- GasUsed: The actual amount of gas consumed
- Status: Whether the transaction succeeded
- Log: Data logged by the transaction

SMART CONTRACT INTERFACES

Goal: Allow to interact with other contracts

- without knowing their implementation
- without knowing their address

Similar to interfaces in object-oriented languages

- Different contracts can implement the same interface
- Contracts implementing the same interface can be used interchangeably

FUNGIBLE TOKENS

What?

- Smart contract that tracks ownership of tokens
- Fungible = interchangeable

Why?

- Manage shares of a company or asset
- Create a custom currency on an existing blockchain

How?

- Keep mapping from address to balance

```
balanceOf: HashMap[address, uint256]
```

FUNGIBLE TOKENS: OPERATIONS

Mint

- Create new tokens
- Not all token contracts allow this

Transfer: Move balance from one address to another

Burn: Remove the specified amount from circulation

Approve:

Give another address an “allowance” to transfer/burn your tokens

TransferFrom/Burn From:

Use your “allowance” to transfer/burn tokens

FUNGIBLE TOKENS: CODE

```
@external
def __init__(_supply: uint256):
    self.balanceOf[msg.sender] = _supply
    self.totalSupply = _supply
    self.minter = msg.sender
```

```
@external
def transfer(_to : address, _value : uint256) -> bool:
    self.balanceOf[msg.sender] -= _value
    self.balanceOf[_to] += _value
```

```
@external
def mint(_to: address, _value: uint256):
    assert msg.sender == self.minter
    assert _to != empty(address)
    self.totalSupply += _value
    self.balanceOf[_to] += _value
```

(Simplified)

EXAMPLE: GAS TOKEN

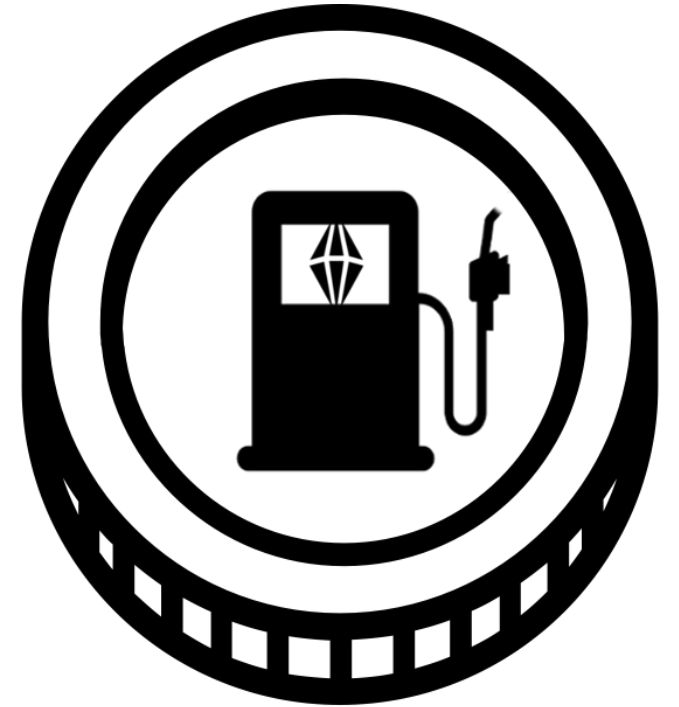
Idea: Store gas in a contract

- Buy gas when it is cheap, sell it when it is expensive?

How?

- Generate gas tokens by allocating data for the contract (will generate gas cost)
- Redeem gas tokens by deleting data (will create a gas refund)

(Unsure if this still works in the most recent version of Ethereum)



gastoken.io/

NON-FUNGIBLE TOKENS

What?

- Smart contract that tracks ownership of tokens
- Non-Fungible = Every token is unique

Why?

- Encode ownership of a physical or virtual item
 - e.g., an item in a video game or a piece of art

How?

- Keep mapping from unique token id to address
- Note: Unlike before, each token as a unique identifier!

```
idToOwner: HashMap[uint256, address]
```

NON-FUNGIBLE TOKENS: OPERATIONS

balanceOf: How many tokens does this account own?

ownerOf: Who owns a particular NFT?

Approve:

Give another address permission to transfer a particular NFT

ApproveAll:

Give another address permission to transfer all my NFTs

TransferFrom:

Transfer a token you own or have an allowance for

NFT EXAMPLE: CRYPTOKITTIES

Each NFT represents a particular “cat”

- Cats can be bought/sold like other NFTs

Cats have DNA (some number of bytes) that define their appearance

- New cats can be “bred” by combining two existing cats
- Image of the NFT is generated from that DNA



NON-FUNGIBLE TOKENS: CODE

```
@external
def __init__():
    self.minter = msg.sender
    self.totalSupply = 0
```

```
@external
def mint(_to: address):
    assert msg.sender == self.minter
    assert _to != ZERO_ADDRESS
    self.totalSupply += 1
    self.idToOwner[self.totalSupply] = _to
```

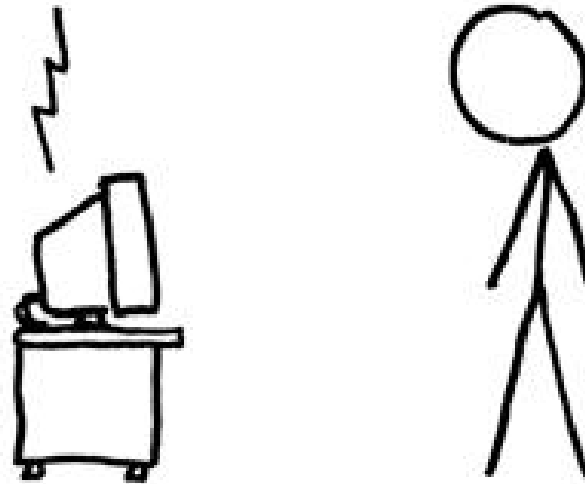
```
@external
def transfer(_to: address, _tokenId: uint256):
    assert _to != ZERO_ADDRESS
    assert self.idToOwner[_tokenId] == msg.sender
    self.idToOwner[_tokenId] = _to
```

(Simplified)

BREAK

WITH THE COLLAPSE OF THE DOLLAR, THE GOVERNMENT
HAS ENDORSED AN ALTERNATE CURRENCY.

YOUR MONETARY WORTH IS NOW DETERMINED BY THE
NUMBER OF FUNNY PICTURES SAVED TO YOUR HARD DRIVE.



I HAVE BEEN PREPARING FOR THIS MOMENT MY WHOLE LIFE.

xkcd predicting NFTs

BLOCKCHAIN GOVERNANCE

How to participate in a blockchain network?

- Run any implementation of the chain *protocol*
- Multiple implementation can (and should!) coexist
- e.g., go-ethereum for Ethereum

Who decides on protocol and code changes?

- Developers have a internal voting process
- e.g., Ethereum Improvement Proposals (EIP)

How do changes get adopted?

- Majority of the network has to migrate to new code
- We talk more about such changes in the future

TOKEN STANDARDS

ERC “**E**thereum **R**equest for **C**omments”

- Application-level EIP
- Defines a **standardized interface** for smart contracts

Why?

- Allows interoperability across smart contracts
- Prevents common errors (interface went through a review process)

ERC-20

Standardized fungible tokens

```
def totalSupply() -> uint256
```

```
def balanceOf(_owner: address) -> uint256
```

```
def allowance(_owner: address, _spender: address) -> uint256
```

```
def transfer(_to: address, _value: uint256) -> bool
```

```
def transferFrom(_from: address, _to: address, _value: uint256) -> bool
```

```
def approve(_spender: address, _value: uint256) -> bool
```

ERC-721

Standardized non-fungible tokens (NFTs)

```
def balanceOf(_owner: address) -> uint256
```

```
def ownerOf(_tokenId: uint256) -> address
```

```
def getApproved(_tokenId: uint256) -> address
```

```
def transferFrom(_from: address, _to: address, _tokenId: uint256)
```

```
def approve(_approved: address, _tokenId: uint256)
```

```
def isApprovedForAll(_owner: address, _operator: address) -> bool
```

```
def setApprovalForAll(_operator: address, _approved: bool)
```

HOW TO BUY CRYPTOCURRENCY

Need some way to trade cryptocurrencies (e.g., exchange BTC for ETH)

Centralized Exchanges

- Controlled by a single entity, e.g., Coinbase Global Inc.
- Can fail at any time

The Coinbase logo, featuring the word "coinbase" in a blue, lowercase, sans-serif font.

DECENTRALIZED EXCHANGES

Idea: Exchange tokens directly on blockchain

How?

- Smart contract facilitates exchange between buyers and sellers
- Blockchain protocol ensures the exchange cannot be tampered with

Limitation:

Can only exchange tokens, not fiat money (e.g., USD)

ORDER BOOKS

Keeps track of sellers and buyers

- Each order has a specific price and quantity

Sell and buy orders need to be matched to succeed

- Buy order presents an upper bound for the price
- Sell order presents a lower bound for the price
- Need to pick a price (and market size) within that range

Why can this be a problem for a decentralized exchange?

- Blockchain transactions may take a long time to confirm
 - Prices may have changed at the before the transaction is confirmed
- Transaction fees may be high and throughput is generally low
 - Representing every order as a transaction is inefficient

Order Book		
Market Size	Price (USD)	My
0.5000	4731.37	
0.2500	4731.31	
0.1037	4731.30	
1.7734	4731.19	
0.5140	4731.15	
0.0934	4731.02	
7.8660	4731.00	
0.5793	4730.90	
0.0363	4730.68	
2.5842	4730.66	
14.4679	4730.65	
29.5260	4730.60	
14.6629	4730.58	
3.2337	4730.57	
3.1721	4730.48	
1.6916	4730.34	
0.9779	4730.26	
2.6097	4730.24	
1.6916	4730.10	
0.4768	4729.94	
0.6165	4729.92	
4.4210	4729.86	
11.4976	4729.77	
0.5113	4729.76	
3.0857	4729.40	
USD Spread		0.01
0.3359	4729.39	
0.1637	4729.38	
2.2202	4729.37	
1.6916	4729.13	
0.2211	4729.07	
0.6633	4729.06	
0.6874	4729.00	
0.4075	4728.82	
0.5852	4728.77	

Coinbase order book for ETH/USD

- Top: Sell offers
- Bottom: Buy offers

MARKET MAKERS

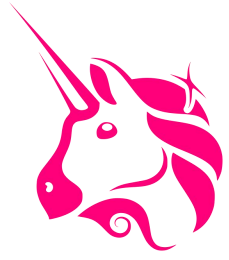


Market makers serves as an intermediate between sellers and buyers

- Market makers can profit from the "ask-bid spread" (difference between sell and buy offers)

Also called "*liquidity provider*"

- They offer a large quantity of the traded item(s) to facilitate continuous trade



UNISWAP

- First proposed by Vitalik Buterin (once again...)¹
- Development started in 2017 by Hayden Adams
- One of the most popular exchanges
- Implemented as a set of Ethereum smart contracts
- Governed by the UNI token, which itself is one of the most traded tokens

¹ <https://vitalik.ca/general/2017/06/22/marketmakers.html>

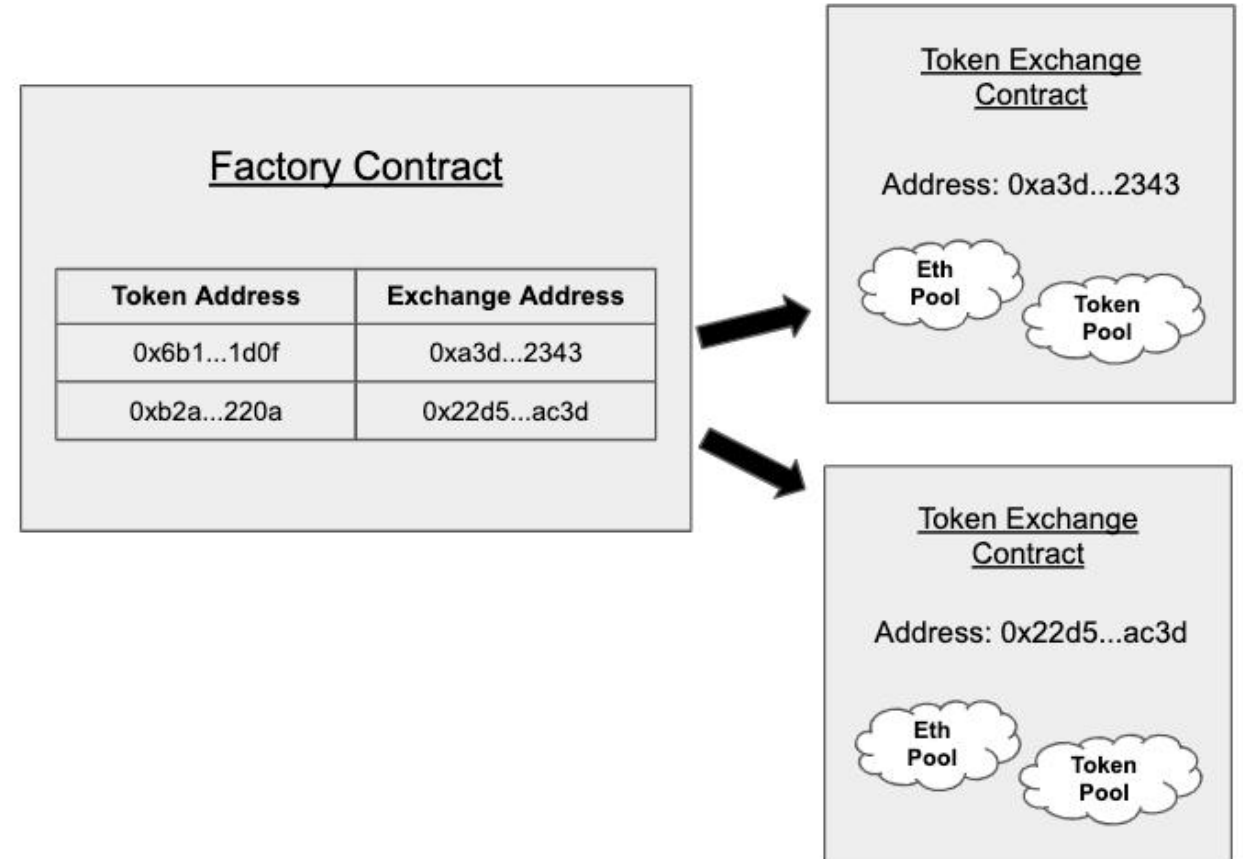
UNISWAP ARCHITECTURE

Factory contract

- Tracks mapping of all existing exchanges
- Creates new exchanges if needed

Exchange contracts

- Facilitate exchanges between Ether and a specific token
- Stores Ether and the token as needed



Source:

<https://docs.ethhub.io/guides/graphical-guide-for-understanding-uniswap/>

LIQUIDITY PROVIDERS IN UNISWAP

- Participants pool currency/tokens in a smart contract
- The contract serves as a liquidity provider
 - One contract per token pair
- When people trade through the contract, a fee is charged
 - Each pool participants gets share of the fee proportional to their pool contribution
- When storing liquidity in an exchange, participants receive some number of exchange-specific tokens in return
- If participants want to exit the exchange, exchange tokens can be converted back into liquidity *at the current exchange rate*

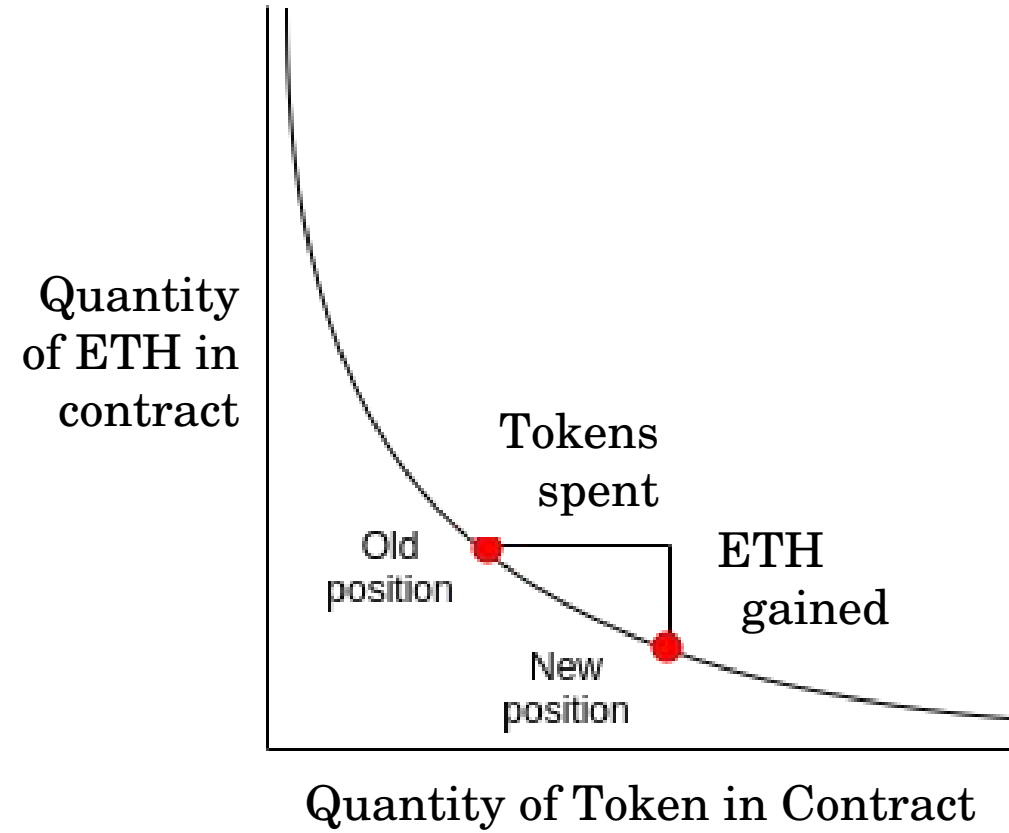
DETERMINING THE PRICE IN UNISWAP (V1)

Price is set by the smart contract so that the following equation always holds:

$$(\text{Amount of Token}) * (\text{Amount of Ether}) = \text{const.}$$

For example

- If contract's supply of Token decreases, the price (in Ether) increases
- If contract's supply of Token increases, the price (in Ether) decreases



Source:

<https://docs.ethhub.io/guides/graphical-guide-for-understanding-uniswap/>

UNISWAP CODE

THAT'S ALL (FOR TODAY)

Next time: Smart contract security