UniswapV2 Opensource Study


윤준성-개발팀-4기


정효영-개발팀-4기

WARNING

저희가 공부하고, 해석한대로 발표에 녹였기 때문에,
사실과 다른 정보가 포함되어 있을 수 있음을 알려드립니다.

질문 및 지적 환영.

Uniswap V2 Opensource Study

·목 표·

유니스왑에 대해 이해한다

코드를 보며, 기능 및 핵심원리를 살펴본다.

유니스왑 코드를 보면서 solidity 실력을 기른다 !

어떤식으로 진행했나요?

서로 각자 맡은 부분을 공부해오고, 공부해온 부분을 만나서
알려주는 방식으로 팀프로젝트를 진행했습니다

목차

- **Uniswap v2 Appearance Background**
- **Price Oracle**
- **Swap**
  - Flash swap
- **Add Liquidity & Remove Liquidity**
  - About LP token
  - LP token의 비영구적 손실
  - Slippage에 관하여 ..

Uniswap v2 Appearance Background

Uniswap V1

- 단일 pool 문제(tokenA → ETH → tokenB)
- 악의적 가격 조작의 가능성
- 추가 기능의 필요성

Uniswap V2

- 직접 pool(tokenA → tokenB) by WETH
- 가격 oracle 개선 → TWAP 이용해 가격 조작 어려움
- Flash Swap 기능 추가

Price Oracle

블록체인에서 Oracle : **블록체인의 스마트 컨트랙트와 오프체인 데이터 공급자를 연결하는 통로** (RDBMS인 Oracle과 블록체인에서 말하는 Oracle은 개념이 다르다.)

Price Oracle

유니스왑V2 Pool이 신뢰할 수 있는 가격정보를 제공해줄수 있다.

->높은 유동성과 차익거래자들로 인해 중앙화된 거래소와 가격이 비슷하다.

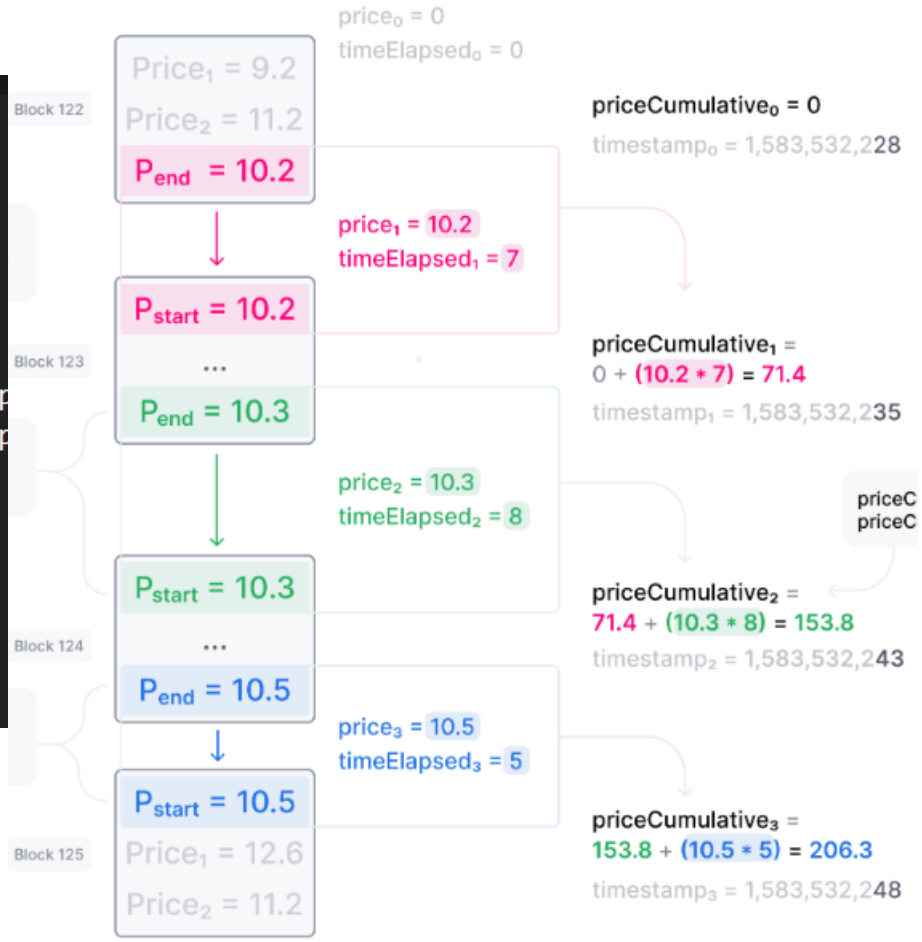
$$a_i = \sum_{i=1}^{t} p_i$$

$$p_{t_1,t_2} = \frac{a_{t_2} - a_{t_1}}{t_2 - t_1}$$

산술평균을 사용해서 Time-weighted average price(TWAP)을 계산 했다. (a는 i초까지 누적 가격)

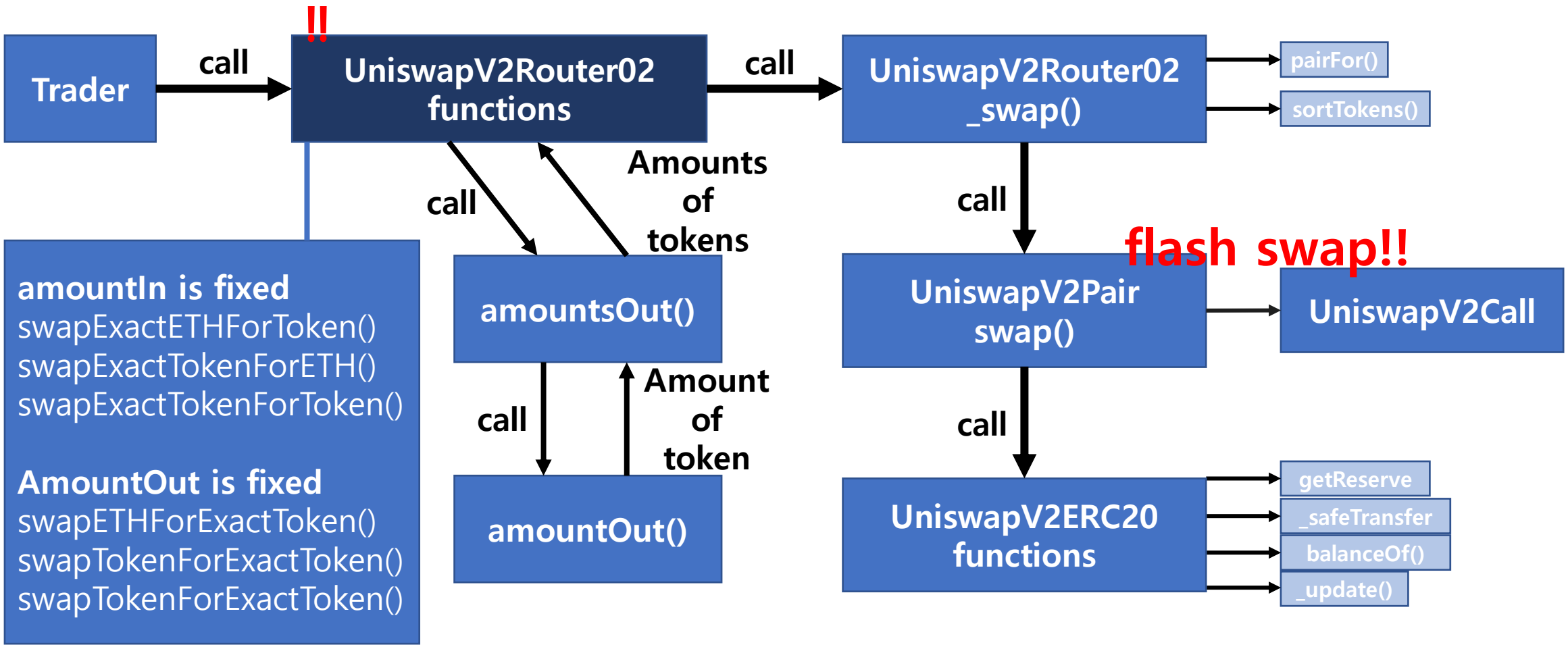
Price Oracle

```
// update reserves and, on the first call per block, price accumulators
function _update(uint balance0, uint balance1, uint112 _reserve0, uint112 _reserve1) private
    require(balance0 <= uint112(-1) && balance1 <= uint112(-1), 'UniswapV2: OVERFLOW');
    uint32 blockTimestamp = uint32(block.timestamp % 2**32);
    uint32 timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired
    if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
        // * never overflows, and + overflow is desired
        price0CumulativeLast += uint(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElap
        price1CumulativeLast += uint(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElap
    }
    reserve0 = uint112(balance0);
    reserve1 = uint112(balance1);
    blockTimestampLast = blockTimestamp;
    emit Sync(reserve0, reserve1);
}
```



전체 계약 내역에서 매초마다 Uniswap 가격의 합을 나타내서 저장 해 놓은 코드 : 이를 이용해서 TWAP를 계산할 수 있다.

Swap


!!
Trader
call
UniswapV2Router02
functions
call
UniswapV2Router02
_swap()
pairFor()
sortTokens()
call
Amounts
of
tokens
amountsOut()
call
Amount
of
token
amountOut()
amountIn is fixed
swapExactETHForToken()
swapExactTokenForETH()
swapExactTokenForToken()
AmountOut is fixed
swapETHForExactToken()
swapTokenForExactToken()
swapTokenForExactToken()
call
flash swap!!
UniswapV2Pair
swap()
UniswapV2Call
call
UniswapV2ERC20
functions
getReserve
_safeTransfer
balanceOf()
_update()

Swap – loop? Path?

```
function _swap(uint[] memory amounts, address[] memory path, address _to) internal virtual {
    for (uint i; i < path.length - 1; i++) {
        (address input, address output) = (path[i], path[i + 1]);
        (address token0, ) = UniswapV2Library.sortTokens(input, output);
        uint amountOut = amounts[i + 1];
        (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
        address to = i < path.length - 2 ? UniswapV2Library.pairFor(factory, output, path[i + 2]) : _to;
        IUniswapV2Pair(UniswapV2Library.pairFor(factory, input, output)).swap(
            amount0Out,
            amount1Out,
            to,
            new bytes(0)
        );
    }
}
```

```
function getAmountsIn(
    uint amountOut,
    address[] memory path
) public view virtual override returns (
    return UniswapV2Library.getAmountsIn
}
```

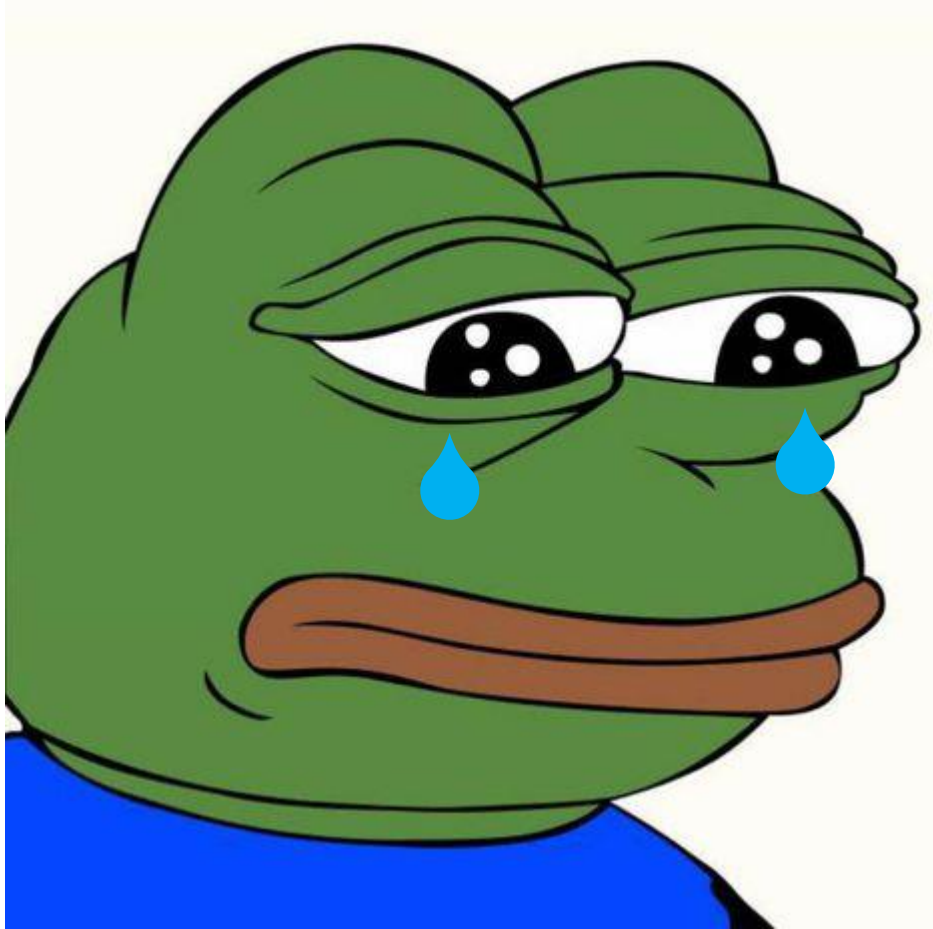
```
function getAmountsOut(
    uint amountIn,
    address[] memory path
) public view virtual override returns (uint[] memory amounts) {
    return UniswapV2Library.getAmountsOut(factory, amountIn, path);
}
```

```
function swapExactTokensForTokens(
    uint amountIn,
    uint amountOutMin,
    address[] calldata path,
    address to,
    uint deadline
) external virtual override ensure(deadline) returns (uint[] memory amounts) {
    amounts = UniswapV2Library.getAmountsOut(factory, amountIn, path);
    require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT');
    TransferHelper.safeTransferFrom(
        path[0],
        msg.sender,
        UniswapV2Library.pairFor(factory, path[0], path[1]),
        amounts[0]
    );
    _swap(amounts, path, to);
}
```

Swap – loop? Path?

ETH to USDT !!

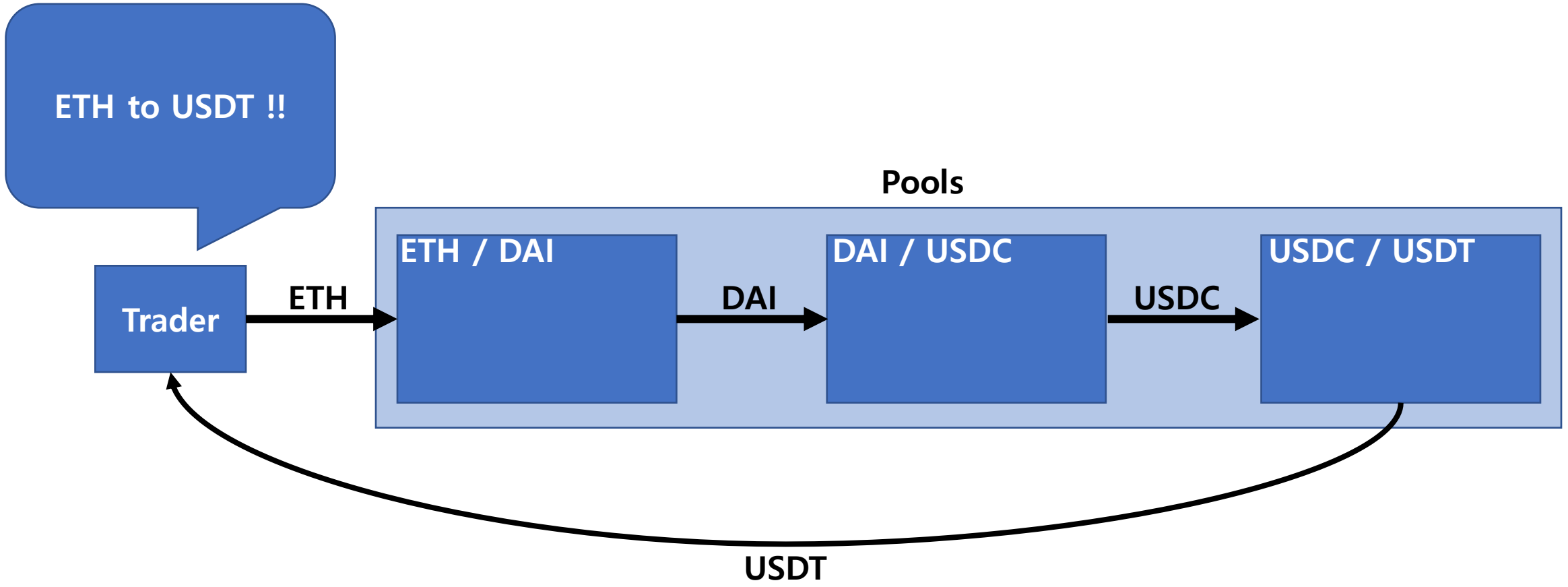
Trader



Swap – loop? Path?

UniswapV2Router!!

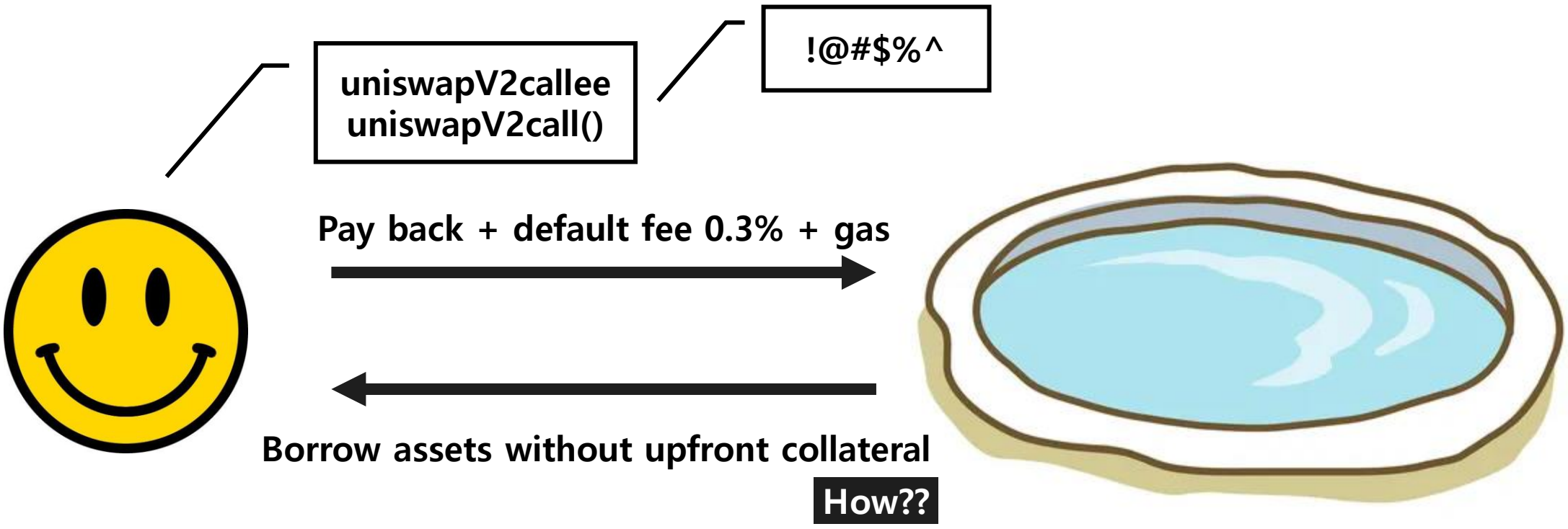
Swap – loop? Path?



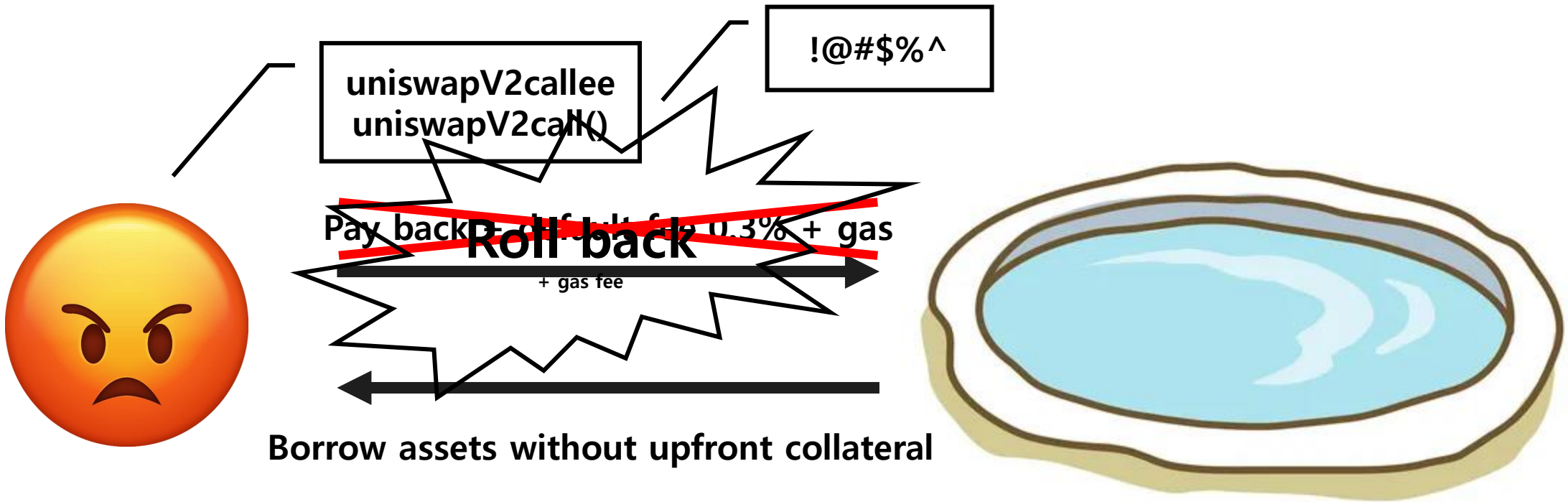
Swap – loop? Path?

```
function _swap(uint[] memory amounts, address[] memory path, address _to) internal virtual {
    for (uint i; i < path.length - 1; i++) {
        (address input, address output) = (path[i], path[i + 1]);
        (address token0, ) = UniswapV2Library.sortTokens(input, output);
        uint amountOut = amounts[i + 1];
        (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
        address to = i < path.length - 2 ? UniswapV2Library.pairFor(factory, output, path[i + 2]) : _to;
        IUniswapV2Pair(UniswapV2Library.pairFor(factory, input, output)).swap(
            amount0Out,
            amount1Out,
            to,
            new bytes(0)
        );
    }
}
```

Flash Swap


uniswapV2callee
uniswapV2call()
!@#$%^
Pay back + default fee 0.3% + gas
Borrow assets without upfront collateral
How??

Flash Swap

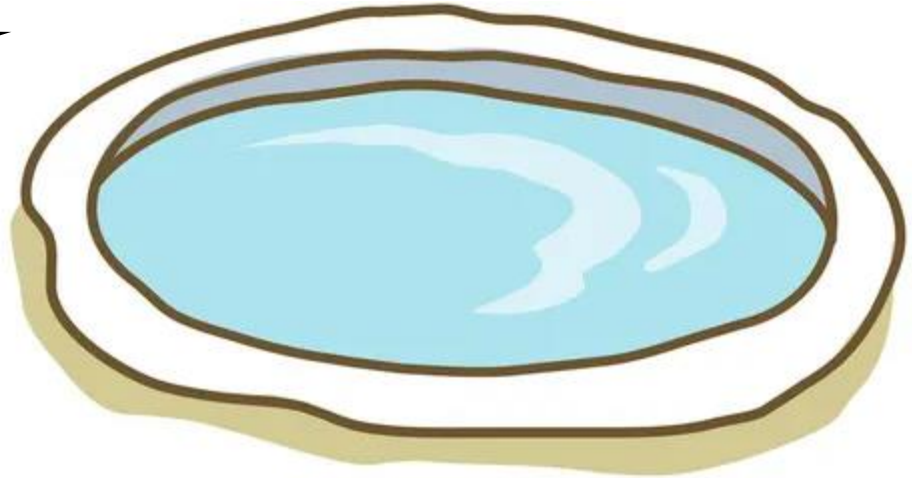


Borrow assets without upfront collateral

Flash Swap

Roll back

+ gas fee



Flash Swap

```
function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data) external lock {
    require(amount0Out > 0 || amount1Out > 0, 'UniswapV2: INSUFFICIENT_OUTPUT_AMOUNT');
    (uint112 _reserve0, uint112 _reserve1, ) = getReserves(); // gas savings
    require(amount0Out < _reserve0 && amount1Out < _reserve1, 'UniswapV2: INSUFFICIENT_LIQUIDITY');

    uint balance0;
    uint balance1;
    {
        // scope for _token{0,1}, avoids stack too deep errors
        address _token0 = token0;
        address _token1 = token1;
        require(to != _token0 && to != _token1, 'UniswapV2: INVALID_TO');
        if (amount0Out > 0) _safeTransfer(_token0, to, amount0Out); // optimistically transfer tokens
        if (amount1Out > 0) _safeTransfer(_token1, to, amount1Out); // optimistically transfer tokens
        if (data.length > 0) IUniswapV2Callee(to).uniswapV2Call(msg.sender, amount0Out, amount1Out, data);
        balance0 = IERC20(_token0).balanceOf(address(this));
        balance1 = IERC20(_token1).balanceOf(address(this));
    }
    uint amount0In = balance0 > _reserve0 - amount0Out ? balance0 - (_reserve0 - amount0Out) : 0;
    uint amount1In = balance1 > _reserve1 - amount1Out ? balance1 - (_reserve1 - amount1Out) : 0;
    require(amount0In > 0 || amount1In > 0, 'UniswapV2: INSUFFICIENT_INPUT_AMOUNT');
    {
        // scope for reserve{0,1}Adjusted, avoids stack too deep errors
        uint balance0Adjusted = balance0.mul(1000).sub(amount0In.mul(3));
        uint balance1Adjusted = balance1.mul(1000).sub(amount1In.mul(3));
        require(
            balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1).mul(1000 ** 2),
            'UniswapV2: K'
        );
    }
```

무담보 대출 실행

Call 함수 호출

내부 구현 로직에서 상환

상환된 금액 계산

상환된 금액 체크

invalid 하다면, revert

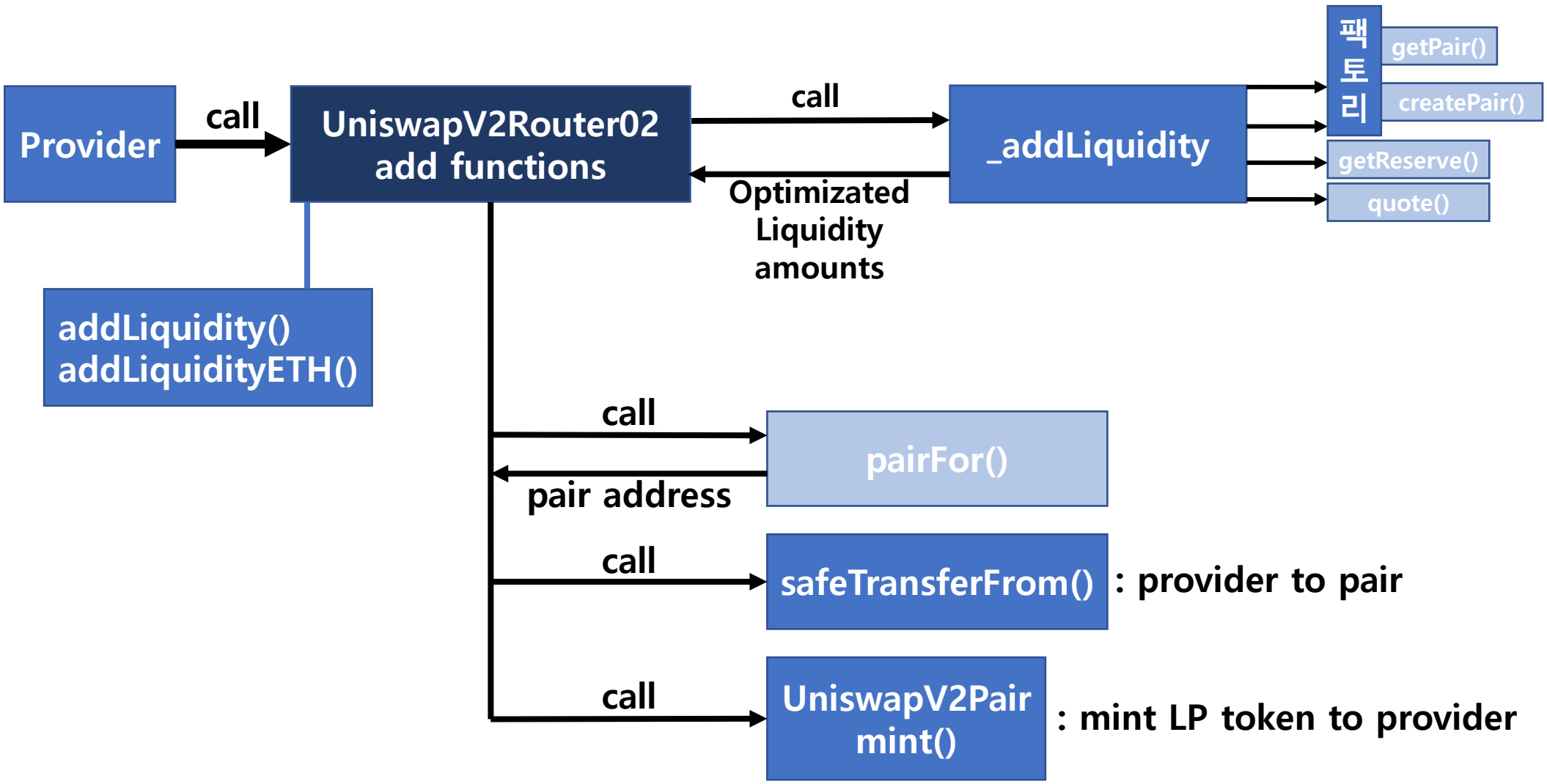
아 힘들다.
Keep going~

헉 ... 그런데 Pool 안에 토큰이 없으면 어떡하지!!!

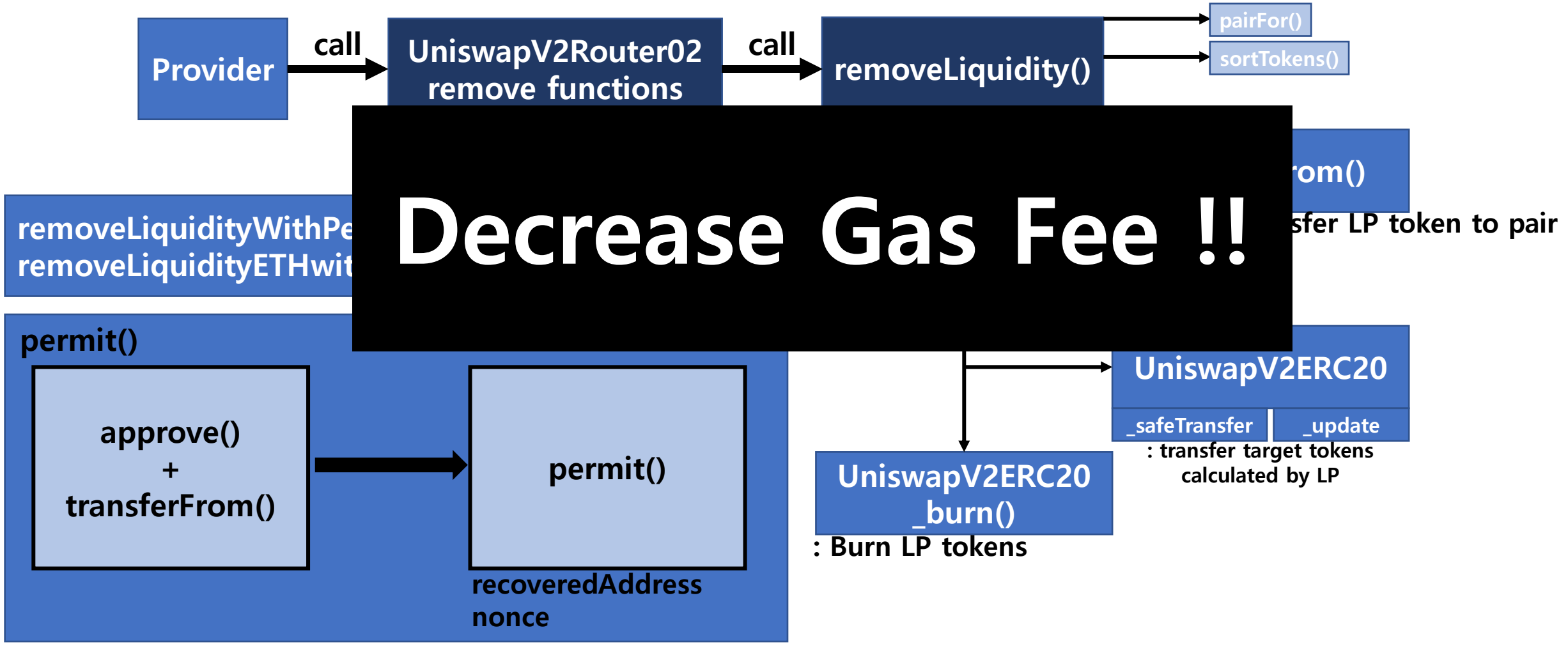
내 이더 딴걸로 바꾸고싶다고 !!!

아! 유동성이라는게 있군 ^^

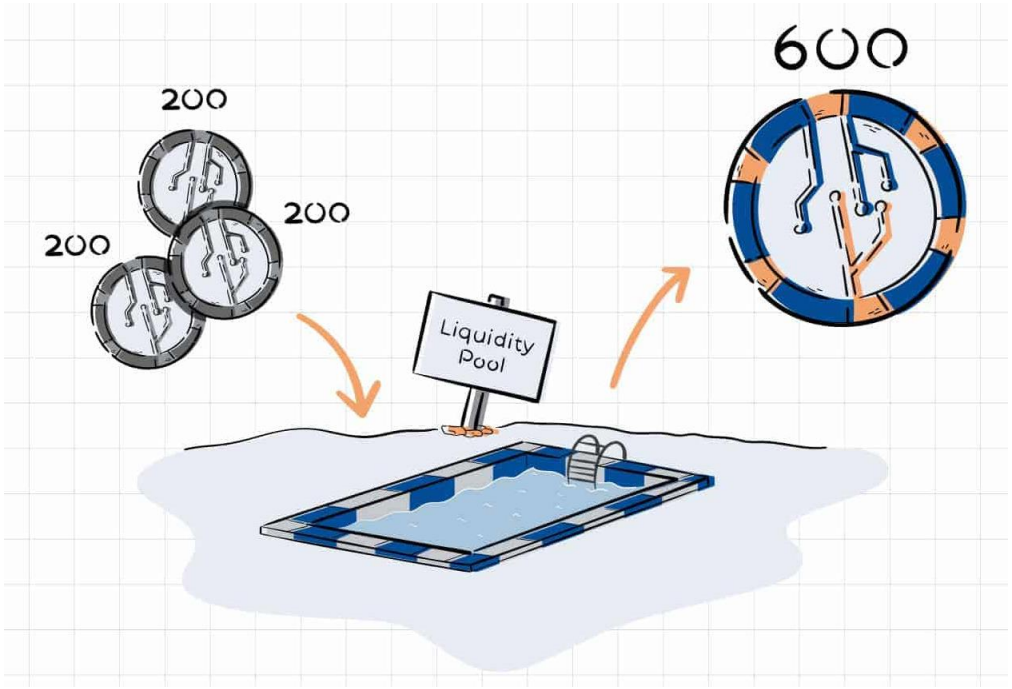
Add Liquidity



Remove Liquidity



About LP

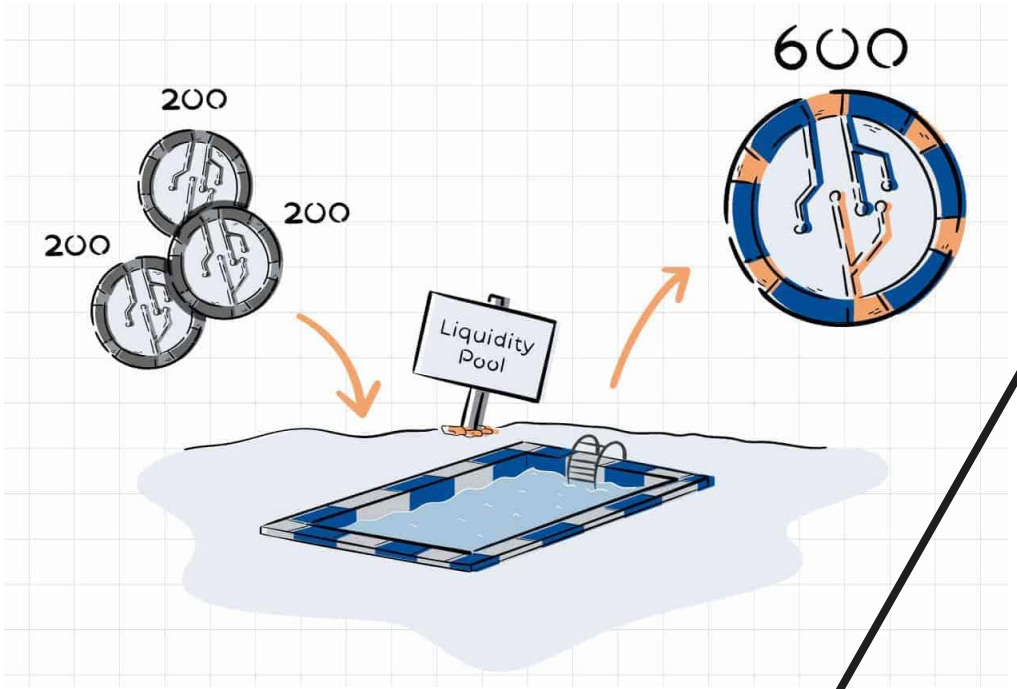



```
liquidity =
Math.min(amount0.mul(_totalSupply) / _reserve0,
amount1.mul(_totalSupply) / _reserve1);
```

최초로 유동성을 생성할 때는 어떨까?

```
if (_totalSupply == 0) {
    liquidity = Math.sqrt(amount0.mul(amount1)).sub(MINIMUM_LIQUIDITY);
    _mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
} else {
    liquidity = Math.min(amount0.mul(_totalSupply) / _reserve0, amount1.mul(_totalSupply) / _reserve1);
}
```

LP – mintFee



$$f_{1,2} = 1 - \frac{\sqrt{k_1}}{\sqrt{k_2}} \quad \begin{pmatrix} k_1 & : \text{이전 유동성} \\ k_2 & : \text{늘어난 유동성} \end{pmatrix}$$

유동성이 늘어나면, 늘어난 만큼의 유동성의 0.05%만큼의 LP를 mint해서 가져간다. (factory contract의 address로 전송)

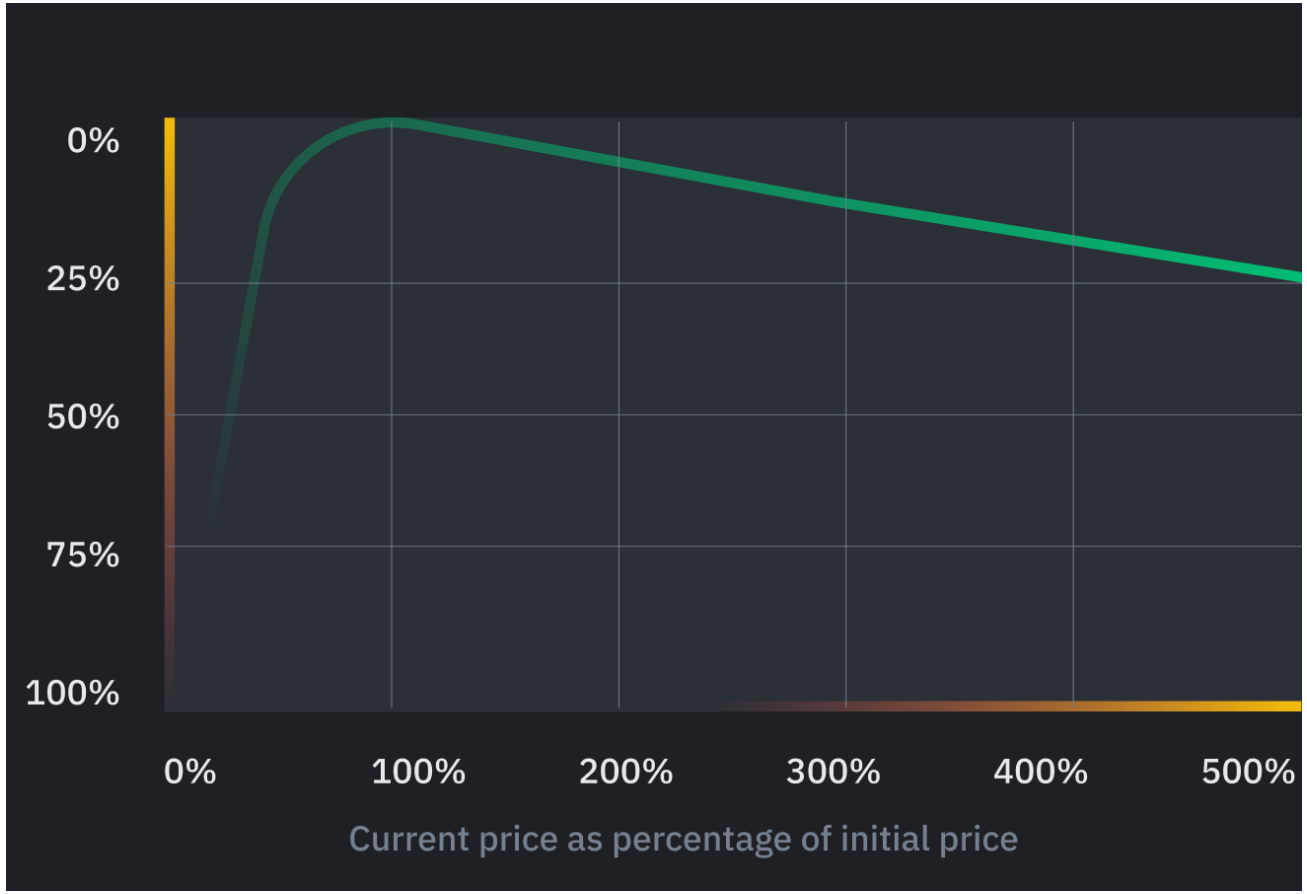
현재는 가져가지 않음!!

```
function setFeeTo(address _feeTo) external {
    require(msg.sender == feeToSetter, 'UniswapV2: FORBIDDEN');
    feeTo = _feeTo;
}
```

```
address feeTo = IUniswapV2Factory(factory).feeTo();
feeOn = feeTo != address(0);
uint _kLast = kLast;
if (feeOn) {
    if (_kLast != 0) {
        uint rootK = Math.sqrt(uint(_reserve0).mul(_reserve1));
        uint rootKLast = Math.sqrt(_kLast);
        if (rootK > rootKLast) {
            uint numerator = totalSupply.mul(rootK.sub(rootKLast));
            uint denominator = rootK.mul(5).add(rootKLast);
            uint liquidity = numerator / denominator;
            if (liquidity > 0) _mint(feeTo, liquidity);
        }
    }
```

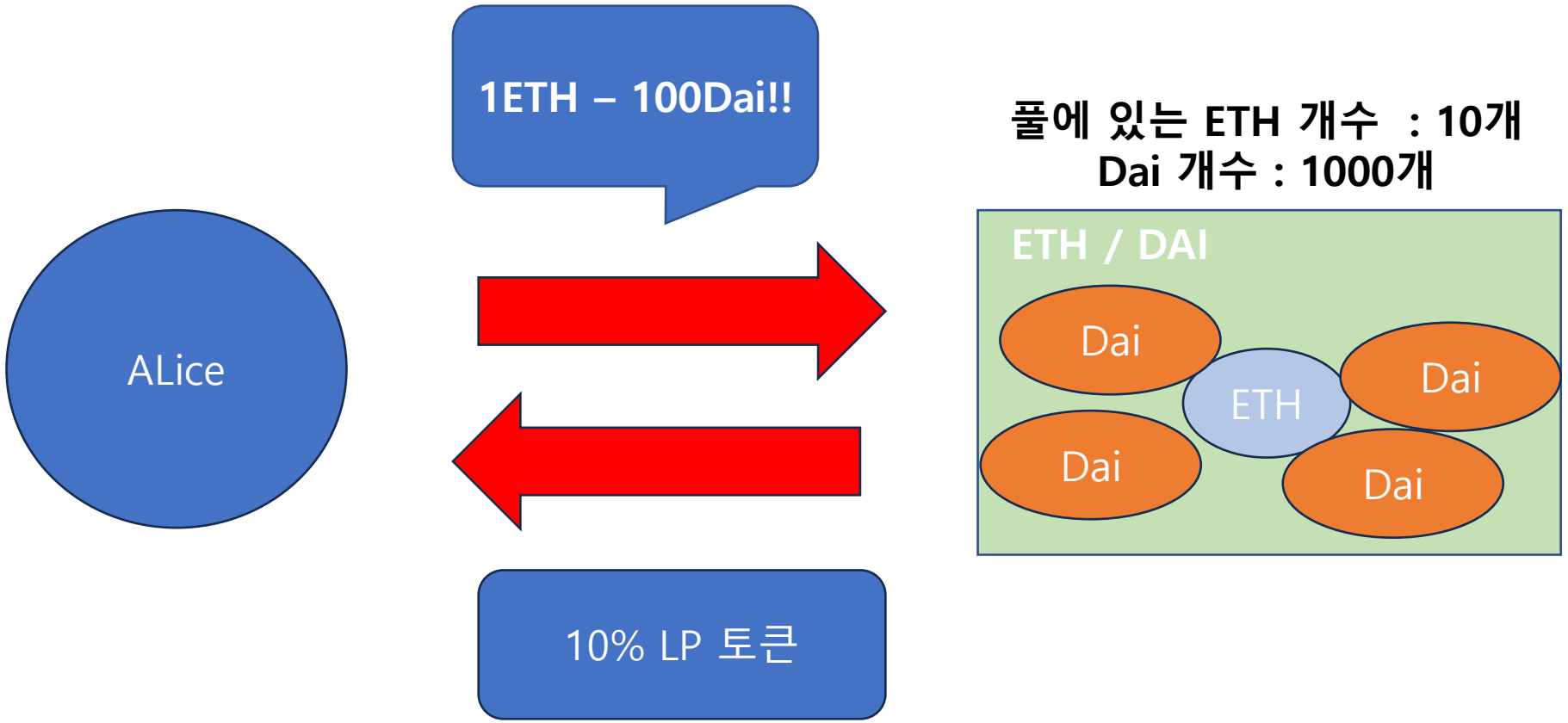
LP - 비영구적 손실

비영구적 손실 : 유동성 풀에 유동성을 공급하고 제거 하였을때, 예치한 자산의 가치가 변동되어 그냥 보유했을 때보다 손해인 상황



-그림은 그냥 자산을 Holding하는것과 유동성 제공한 것을비교하여 생기는 손실을 보여줌

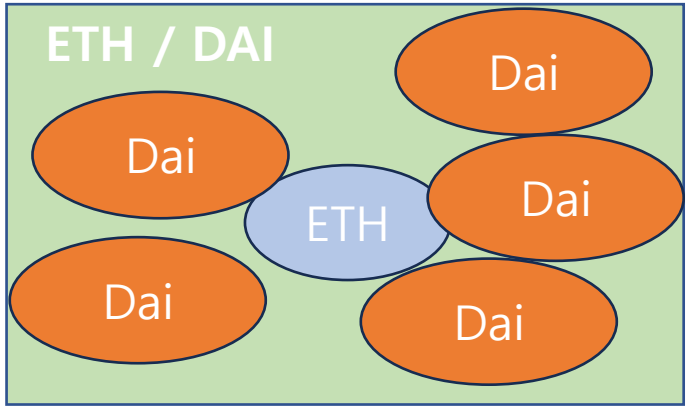
LP - 비영구적 손실


1ETH – 100Dai!!
ALice
10% LP 토큰
풀에 있는 ETH 개수 : 10개
Dai 개수 : 1000개
ETH / DAI
Dai
ETH
Dai
Dai
Dai

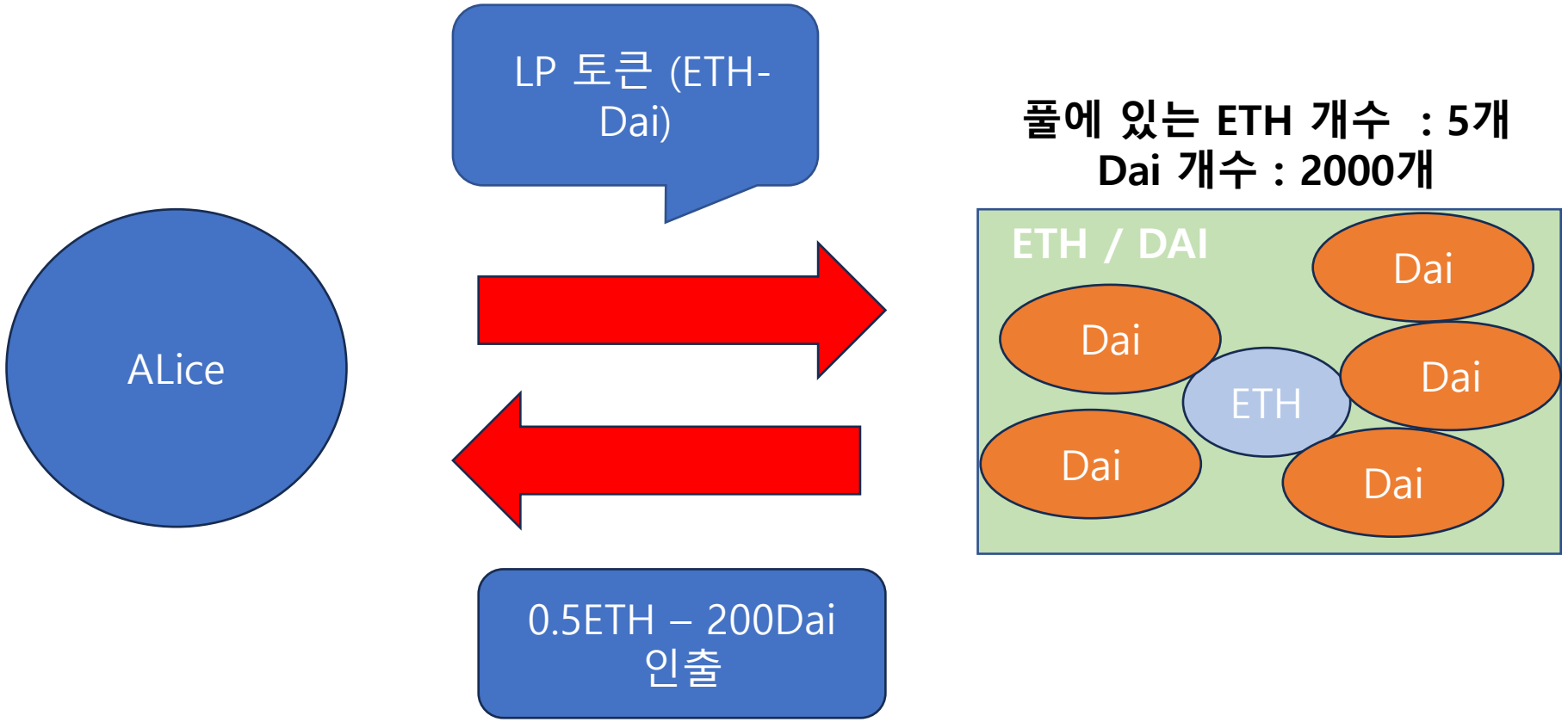
LP - 비영구적 손실

1 ETH가격 = 400Dai

풀에 있는 ETH 개수 : 5개
Dai 개수 : 2000개



LP - 비영구적 손실


LP 토큰 (ETH-Dai)
ALice
0.5ETH – 200Dai 인출
풀에 있는 ETH 개수 : 5개
Dai 개수 : 2000개
ETH / DAI
Dai
Dai
Dai
ETH
Dai
Dai

LP - 비영구적 손실

	ETH	Dai	총자산
초기자산	1ETH = 100$	100Dai = 100$	200$
ETH 떡상 후 자산	0.5 ETH = 200$	200Dai = 200$	400$
Holding 했을시 예상 자산	1ETH = 400$	100Dai = 100$	500$

-그냥 Hodling 했을 때에 비해 100$의 손실이 일어 난다.
-Uniswap 0.3%보상등으로 손실을 메꿔야함

Slippage

Slippage?

매매 주문 시 체결 오차가 발생하여 원하는 가격과 다른 가격으로 실제 거래가 체결되었을 때의 가격 차이

Transaction서명과 finalize의 시간 차이

Slippage

Impact 1

The greater the liquidity, the smaller the price impact

Impact 2

The larger the number of tokens you want to swap, the greater the price impact

되짚어보기

- Price oracle을 살펴보며, token의 가격이 어떻게 결정되는지 알아보았어요.
- Swap의 동작이 어떻게 일어나는지, 함수 호출 관계를 정리해보며 알아보았고, 관련 핵심 기능인 flash swap에 대해 알아보았어요
- Add, Remove Liquidity의 동작이 어떻게 일어나는지 함수 호출 관계를 정리해보며 알아보았고, 관련 핵심 기능인 LP 교환, 그리고 핵심 개념인 비영구적 손실과 Slippage에 대해 알아보았어요.

QnA

질문이 있으신가요?

없겠죠 .. 네 ..

그럴 일은 없겠지만 혹시 발표 자료가 필요하시다면 슬랙 디엠 주세요 ... ^^

발표 들어주신다고 수고하셧음미다 ^^