https://github.com/0xFLIPSY/CryptoPhunksMarket

CryptoPhunksMarket

Issue 1 (fixed)

In the constructor, there can be a more effective way of doing a check that the phunksContract set is an ERC721 contract rather than just checking for the zero address.

E.g.
IERC721(initialPhunksAddress).balanceOf(address(this));

Issue 2

withdraw function will fail if the seller is the following:

- Contract that does not implement fallback function
- Contract that has fallback function which costs more than 2300 gas

It would be better to send as weth instead of eth if the destination address is a contract.

Issue 3 (fixed)

Funds can be temporarily stuck for bids where the NFT's ownership is transferred to the bidder after a bid is done.
This is because of this line in withdrawBidForPhunk:
if (phunksContract.ownerOf(phunkIndex) == msg.sender) revert();

E.g
Bidder bids on NFT id 1.
Seller sends NFT id 1 to bidder.
Bidder is now owner of id 1, and cannot withdraw his bid unless he transfers id 1 to another address

Consider removing that check as the check is already done in the bid function

Issue 4

All public functions that are used can be changed to external if never called within the same function.

Issue 5 (fixed)

Inconsistent behavior between offerPhunkForSale and offerPhunkForSaleToAddress.
if (phunksContract.getApproved(phunkIndex) != address(this)) revert(); is done in the latter, but not the former.

Issue 6 (fixed)

It is possible for seller of NFT to buy the NFT using buyPhunk as there is no check that seller != msg.sender.

Issue 7 (fixed)

Add nonReentrant to all external facing functions as extra precaution against reentrancy attacks.

Issue 8 (fixed)

The logic for checking if msg.value is valid is wrong:
```
 if (msg.value < offer.minValue) revert();
```

In the dapp UI, the price shown is the offer.minValue, and there is no option to purchase at a value greater than the shown price. However, here in buyPhunk, the msg.value can be greater than the minValue, which does not match the functionality at the UI level.

The proper check should be the following:
```
 if (msg.value != offer.minValue) revert();
```

Issue 9 (Fixed)

Check for ownerOf(phunkIndex) == address(0x0) is unnecessary.
Already done in ERC721.

```
67      /**
68       * @dev See {IERC721-ownerOf}.
69       */
70      function ownerOf(uint256 tokenId) public view virtual override returns (address) {
71          address owner = _owners[tokenId];
72          require(owner != address(0), "ERC721: owner query for nonexistent token");
73          return owner;
74      }
75
```

Issue 10

Griefing of bids can be done by the following:
A is a real bidder who bids 1 ETH on id 1.
B is a griefer who bids 1.1ETH on id 1, thus overwriting A's bid.
B then cancels the bid by calling withdrawBidForPhunk. There is no active bid for id 1.

This is not a simple issue to solve, as removing withdrawBidForPhunk means that the bidder's funds are stuck in the contract unless the seller accepts it.

Issue 11 (fixed)

Inconsistent behavior between buyPhunk and acceptBidForPhunk for taking off a phunk for sale after it has been sold.

buyPhunk
phunkNoLongerForSale(phunkIndex);

acceptBidForPhunk
phunksOfferedForSale[phunkIndex] = Offer(false, phunkIndex, bidder, 0, address(0x0));

The behavior in the two functions should be standardized.

phunksOfferedForSale[phunkIndex] = Offer(false, phunkIndex, msg.sender, 0, address(0x0));

Issue 12 (fixed)

Check effects interactions pattern not followed in buyPhunk

As-is:

phunksContract.safeTransferFrom(seller, msg.sender, phunkIndex);

```
        phunkNoLongerForSale(phunkIndex);
        pendingWithdrawals[seller] += msg.value;
        emit PhunkBought(phunkIndex, msg.value, seller, msg.sender);
```

To-be:

```
        phunksOfferedForSale[phunkIndex] = Offer(false, phunkIndex, msg.sender, 0,
address(0x0));
        phunksContract.safeTransferFrom(seller, msg.sender, phunkIndex);
        pendingWithdrawals[seller] += msg.value;
        emit PhunkBought(phunkIndex, msg.value, seller, msg.sender);
```

Issue 13 (fixed)

Check effects interactions pattern not followed in acceptBidForPhunk

As-is:

```
        phunksContract.safeTransferFrom(msg.sender, bidder, phunkIndex);
        phunksOfferedForSale[phunkIndex] = Offer(false, phunkIndex, bidder, 0, address(0x0));
        uint amount = bid.value;
        phunkBids[phunkIndex] = Bid(false, phunkIndex, address(0x0), 0);
        pendingWithdrawals[seller] += amount;
        emit PhunkBought(phunkIndex, bid.value, seller, bidder);
```

To-be:

```
        phunksOfferedForSale[phunkIndex] = Offer(false, phunkIndex, bidder, 0, address(0x0));
        uint amount = bid.value;
        phunkBids[phunkIndex] = Bid(false, phunkIndex, address(0x0), 0);
        phunksContract.safeTransferFrom(msg.sender, bidder, phunkIndex);
        pendingWithdrawals[seller] += amount;
        emit PhunkBought(phunkIndex, bid.value, seller, bidder);
```

Issue 14 (fixed)

It is possible for the bidder to be the same as the owner of the NFT.
Address A bids for id 1 owned by address B
Address B sends id 1 to address A. Address A is now the owner of id 1
Address A calls acceptBidForPhunk. A is the bidder and the seller.

Add the following check in acceptBidForPhunk that seller != bidder.

Issue 15

Active bids can be stuck and cannot be withdrawn if setPhunksContract is used to change phunksContract. This is due to this line where the returned address of the ownerOf function call is the zero address, or reverts due to incompatible/unimplemented interface.

```
if (phunksContract.ownerOf(phunkIndex) == address(0x0)) revert();
if (phunksContract.ownerOf(phunkIndex) == msg.sender) revert();
```

CryptoPhunksV2

Issue 1

Remove useRandomAvailableToken as usage for it has been commented out.

Issue 2

seedFreeRolls only checks the first item in the array

```
 require(numOfFreeRolls[0] <= 3, "cannot give more than 3 free rolls");
```

Should check for all items.

```
    for (uint256 i = 0; i < tokenOwners.length; i++) {
      require(numOfFreeRolls[i] <= 3, "cannot give more than 3 free rolls");
      freeRollPhunks[tokenOwners[i]] = numOfFreeRolls[i];
    }
```

Issue 3

The seeded free rolls in seedFreeRolls do not count to the 200 limit enforced in giveFreeRoll. Please confirm if this is intended.

Issue 4

Pricing logic in getCostForMintingPhunks looks wrong

```
    if (_numToMint == 1) {
      return 0.02 ether;
```

```
  } else if (_numToMint == 3) {
    return 0.05 ether;
  } else if (_numToMint == 5) {
    return 0.07 ether;
  } else if (_numToMint == 10) {
    return 0.10 ether;
  } else {
    revert("Unsupported mint amount");
  }
```

If you try to mint any number of NFTs other than 1,3,5 or 10, it will fail. For example, 2. Shouldn’t it be <= instead of ==?