uToken Documentation

A plugin to bridge Unity to Ethereum.

Ethereum

Connect

```
string network = "mainnet"; // mainnet ropsten kovan rinkeby goerli
Ethereum ethereum = new Ethereum(network);
```

Block Height

```
BigInteger blockHeight = await ethereum.BlockHeight();
print(blockHeight);
```

Balance Of

```
string account  = "0xaca9b6d9b1636d99156bb12825c75de1e5a58870";
BigInteger wei = await ethereum.BalanceOf(account);
print("wei: " + wei);
print("eth: " + Web3.Convert.FromWei(wei));
```

Send

If WalletScene is not being used, then a private key is required.

```
string privateKey = "0000000000000000000000000000000000000000000000000000000000000001";
Ethereum ethereum = new Ethereum(network, privateKey);

string toAccount = "0x72b8Df71072E38E8548F9565A322B04b9C752932";
decimal ethAmount = 0.01m;

string transactionHash = await ethereum.Send(toAccount, ethAmount);
print(transactionHash);
```

Wallet

Public and private key management can either be done through the WalletScene or manually.

WalletScene

Add `/uToken/Scenes/WalletScene` to the beginning of your build settings. To access wallet scene private key and account:

```
string password = ""; // default is empty
string privateKey = UnityWallet.PrivateKey(password);
print(privateKey);

string account = UnityWallet.Account();
print(account);
```

Manual

To generate your own account.

```
using System;
using NBitcoin;
using Nethereum.HdWallet;
using Nethereum.KeyStore.Model;

void GenerateWallet()
{
  // generate mneumonic
  Mnemonic mnemo = new Mnemonic(Wordlist.English, WordCount.Twelve);
  // create new wallet
  var wallet = new Wallet(mnemo.ToString(), "");
  // access Hierarchical Deterministic wallet 0
  var account = wallet.GetAccount(0);

  // encrypt account with password
  string password = "";
  var keyStoreService = new Nethereum.KeyStore.KeyStoreScryptService();
  var scryptParams = new ScryptParams { Dklen = 32, N = 262144, R = 1, P = 8 };
  var ecKey = new Nethereum.Signer.EthECKey(account.PrivateKey);
  var keyStore = keyStoreService.EncryptAndGenerateKeyStore(password, ecKey.GetPrivateKeyAsB
  string json = keyStoreService.SerializeKeyStoreToJson(keyStore);

  // decrypt account
  print("account: " + account.Address);
  string decryptedPrivateKey = BitConverter.ToString(keyStoreService.DecryptKeyStoreFromJson
  print("decrypted private key: " + decryptedPrivateKey);
}
```