Winner Winner Chicken Dinner

Gauravdeep Singh Bindra

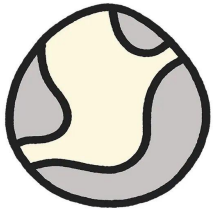
Overview

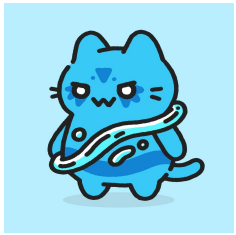
Web3 game

Users get to interact with and influence the state of the “game”

Inspired by Tamagotchi - digital pet launched in Japan with whom you can interact

Relevance to Blockchain





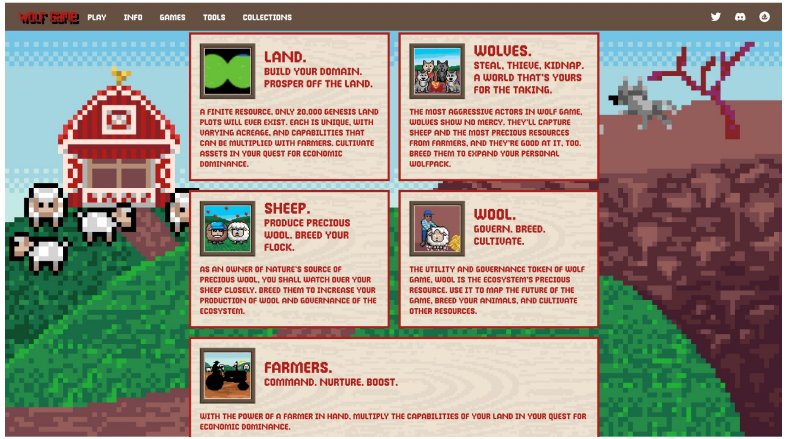
Inspired by Cool Cats NFT project

Collective “meme” sharing

The status of the game can represent the status of a blockchain project community, crypto prices etc

(Future scope: Build a raffle into it where the odds are determined by the status of the game)

Original idea



Build a web3 game that brings together and applies all concepts we learned in class

- Staking
- NFTs
- Randomization
- (+game theory)

Inspired by wolfgame

Factions - stake coins - take decisions - prisoners dilemma - levels - winner(s) get all the staked coins

OBJECTIVES AND ARCHITECTURE

Deploying the smart contract connected to the front end

Anyone should be able to interact with it

The smart contract controls the state of the game(emoji) that is visible to everyone

FRAMERWORKS USED

`Foundry -` Ethereum application development toolkit designed for efficiency and speed in smart contract development

`Svelte kit -` development framework designed to build efficient and fast web applications. Used for Wallet connect

Chainlink keepers

Used Chainlink keepers to keep track of time to automatically reduce the attribute values after a given time. provides a reliable way to automate smart contract triggers and maintenance tasks on the blockchain.

Registered upkeep on chainlink

Upkeeps work with LINK. Got Testnet LINK from LINK faucet.

Explanation of Chainlink Keepers(from ChatGPT)

How Chainlink Keepers Work:

1. **Registering a Job**: A developer or a smart contract registers a job with the Chainlink Keeper network. This involves defining a smart contract that includes two specific functions:
   - `checkUpkeep`: This function returns a boolean indicating whether the upkeep should be performed (`true`), and it can optionally include data that should be sent to the `performUpkeep` function.
   - `performUpkeep`: This function is executed when `checkUpkeep` returns `true`, and it contains the logic of the maintenance task or trigger.
2. **Funding**: The smart contract must be funded with LINK tokens, which are used to pay for the execution of upkeep tasks.
3. **Execution by Keepers**: Registered Keeper nodes constantly monitor registered jobs. When they detect that the `checkUpkeep` condition returns `true`, they execute the `performUpkeep` function. The Keepers are incentivized by receiving LINK tokens in exchange for their work.
4. **Verification and Transparency**: All actions taken by the Keepers are recorded on the blockchain, ensuring transparency and verifiability of the automated tasks.

EMOJI IS AN SVG FILE AND NOT AN IMAGE

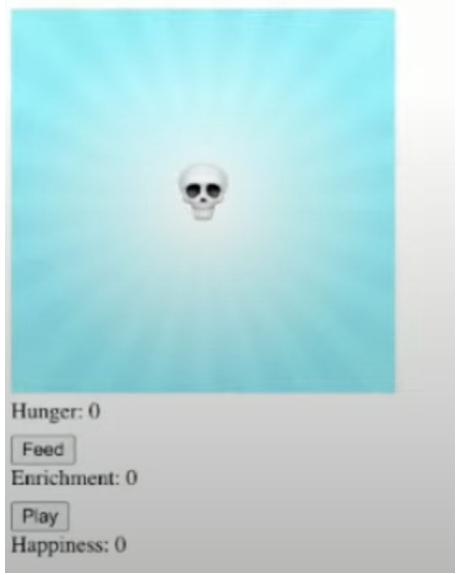
Happiness, Hunger, Enrichment - Three attributes. Happiness is based on Hunger and Enrichment scores.

Image - the image is an NFT - on chain - wanted to have it fully on chain - SVG file - base64 from OpenZepplin - browser renders the SVG out since it is base64 encoded. So this is basically a meta game - the NFT is itself the game

The SVG is updated based on : Checked (Time?) - last time, time was checked
Checked = block.timestamp - time stamp of the block.

Implementation & Evaluation

Working link:
https://drive.google.com/file/d/1SBW0RXvPM_lhWSvUyPz_fDNO0qFBnPxz/view?usp=sharing



Challenges and Lessons

Always work in a team

Since i didnt have experience with coding in Solidity, should have chosen a simpler project from the start and probably one where there are some resources available to refer and get started.

References

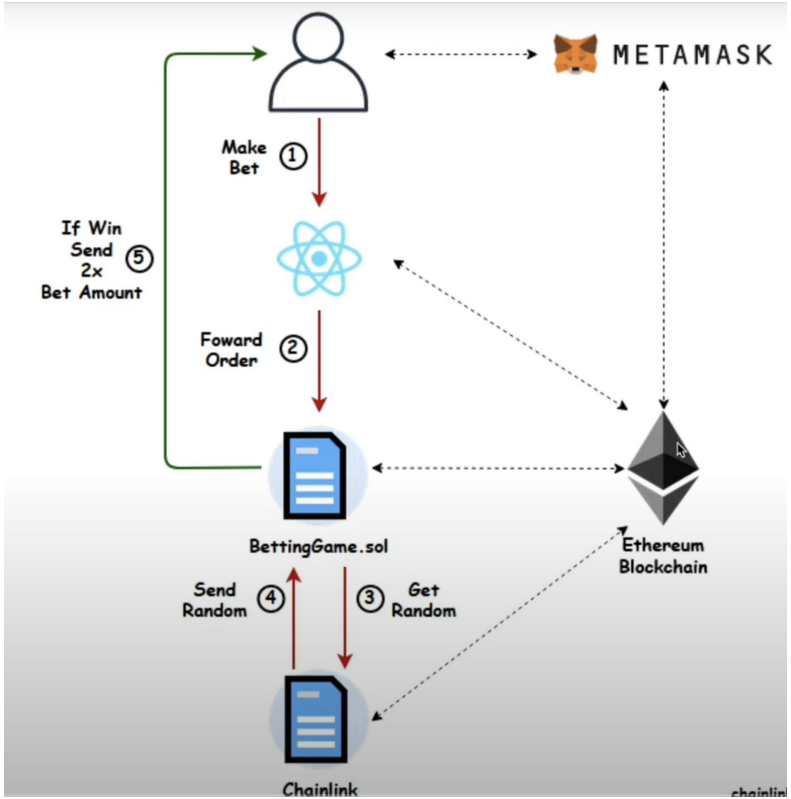
Managing upkeep : https://docs.chain.link/chainlink-automation/guides/manage-upkeeps

Foundry: https://book.getfoundry.sh/

SvelteKit: https://kit.svelte.dev/

Winner Winner Chicken Dinner - 2 Chainlink betting game

ARCHITECTURE



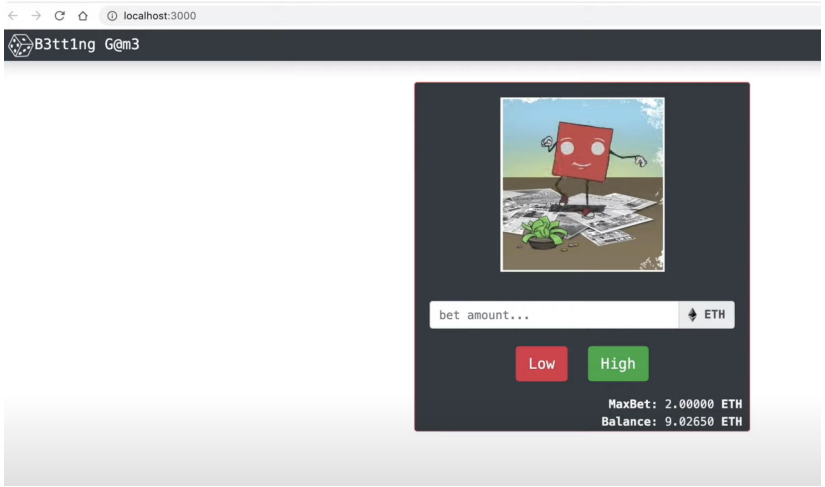
Main Technology

Chainlink Protocol - for randomness of the dice-roll

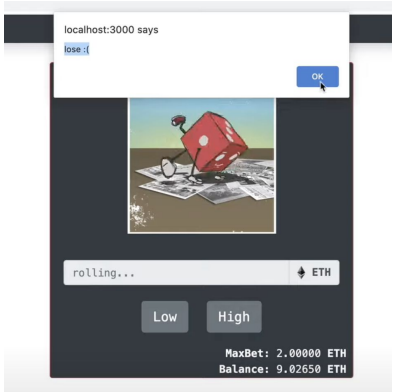
Chainlink is an oracle service

We need Chainlink because of the problems with being able to generate random numbers like we discussed in class.

Front-end



This is how the front-end will look. MaxBet is based on the ETH in the smart contact while Balance is the balance in user's wallet that is connected to the website.



Pop-up showing the result followed by the transaction and account update

Working

Smart contract is loaded with some Sepolia ETH and LINK.
Needs LINK to pay to Chainlink smart contract - to provide random number to our smart contract.

The ETH in the SmartContract determines the current max bet amount. Users can bet only in ETH. No other crypto currency supported.

Users can bet on Low/High. Low covers dice roll values(1,2,3) and High covers dice roll values(4,5,6)

If users bet is placed correctly, they get 2X their bet amount back, otherwise they get nothing back. (In a full-fledged application ofcourse they will get <2X because of some margin for the Casino/SmartContract creator.

References

ChainlinkVRF : https://docs.chain.link/vrf/v2/direct-funding
https://vrf.chain.link/

Getting a random number on Chainlink:
https://docs.chain.link/vrf/v2/subscription/examples/get-a-random-number

Sepolia Faucet for chainlink:
https://faucets.chain.link/sepolia

Chainlink repo and documentation:https://github.com/smartcontractkit/chainlink/blob/develop/contracts/src/v0.8/vrf/VRFConsumerBaseV2.sol

Thanks!