Plinko

Plinko is an example implementation of the popular casino game. The Plinko game on Sui incorporates advanced cryptographic techniques to ensure fairness and transparency. Players drop Plinko balls onto a pegged board, where they randomly fall into slots representing different multipliers. This document details the game's mechanics, cryptographic features, and the methodology for calculating trace paths and verifying signatures.

Building an on-chain Plinko game shares a lot of similarities with the Coin Flip game and Blackjack game. For that reason, this example covers only the smart contracts (Move modules) and frontend logic.

You can find the source files for this example in the Plinko repo on GitHub.

A version of the game is also deployed at Mysten Plinko .

The Plinko game, implemented through smart contracts on the Sui blockchain, incorporates cryptographic techniques to ensure fairness and transparency. Utilizing a blend of BLS signatures, hash functions, and verifiable random function (VRF) inputs, the game calculates the trace path for each ball, determining the game's outcome based on the number of Plink balls a player chooses to drop.

The game mechanics involve a player starting a game by specifying the number of balls and staking a certain amount. The backend generates a BLS signature for the game's randomness source, which is verified on-chain to ensure it's untampered. The game uses a Counter NFT for each round to generate a unique VRF input, ensuring each game's randomness is distinct and cannot be predicted or repeated. The number of Plinko balls, chosen by the player, directly influences the game's complexity and potential payout, as each ball's final position is determined by traversing a cryptographic trace path generated from the hashed BLS signature extended to accommodate the total number of balls.

Follow the comments in each module's code to understand the logic each creates.

The plinko::plinko module combines various Sui blockchain features, such as coin handling, event emissions, and dynamic object fields, to create on-chain Plinko games.

Error handling is integral to the module, with specific codes indicating various failure states or invalid operations:

Provide read-only access to the game's properties, such as:

Include utilities like:

The plinko::house_data module in the Plinko game is designed to manage the game's treasury and configurations. It's responsible for storing the house funds, setting the game parameters (like maximum and minimum stakes), and handling game fees. It also stores the house public key for verifying game outcomes. The module provides functions to adjust game settings, manage the house funds, and ensure the integrity and fairness of the game through cryptographic verification.

The module defines specific error codes to handle exceptional scenarios:

Provide read-only and mutable access to house data properties, enabling operations like querying balance, stake limits, fees, and modifying configurations within authorized contexts:

The plinko::counter_nft module introduces a unique, non-transferable Counter NFT. This Counter object is pivotal for generating a distinct VRF input for each round of the game, thereby ensuring the randomness and fairness of game outcomes. It includes functionalities to create, increment, and destroy these Counter objects. Each Counter is tied to a unique game round and increments its count with each use, providing a fresh input for the game's randomness mechanism.

Unlike traditional NFTs, the Counter NFT is non-transferable, ensuring it remains tied to its original owner and serves its purpose without the risk of duplication or unauthorized transfer.

Navigate to the setup folder of the Plinko repository and execute the publish.sh script. Refer to the README instructions for deploying the smart contracts and testing them locally.

The Plinko component, a central element of the Plinko game's frontend module, is structured to create an interactive and responsive gaming experience. Written in React, it integrates several features and functions to handle the game's logic and user interactions effectively.

State hooks: The module uses React's useState to manage various game states, including finalPaths , isPlaying , totalWon , and more. These states are vital for tracking the current game status and updating the UI accordingly.

Modal for game initialization: A modal is implemented to facilitate the creation of a new game. It captures the randomness and uses handlePlayClick to initiate the game with the backend.

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

Gameplay

The Plinko game, implemented through smart contracts on the Sui blockchain, incorporates cryptographic techniques to ensure fairness and transparency. Utilizing a blend of BLS signatures, hash functions, and verifiable random function (VRF) inputs, the game calculates the trace path for each ball, determining the game's outcome based on the number of Plink balls a player chooses to drop.

The game mechanics involve a player starting a game by specifying the number of balls and staking a certain amount. The backend generates a BLS signature for the game's randomness source, which is verified on-chain to ensure it's untampered. The game uses a Counter NFT for each round to generate a unique VRF input, ensuring each game's randomness is distinct and cannot be predicted or repeated. The number of Plinko balls, chosen by the player, directly influences the game's complexity and potential payout, as each ball's final position is determined by traversing a cryptographic trace path generated from the hashed BLS signature extended to accommodate the total number of balls.

Follow the comments in each module's code to understand the logic each creates.

The plinko::plinko module combines various Sui blockchain features, such as coin handling, event emissions, and dynamic object fields, to create on-chain Plinko games.

Error handling is integral to the module, with specific codes indicating various failure states or invalid operations:

Provide read-only access to the game's properties, such as:

Include utilities like:

The plinko::house_data module in the Plinko game is designed to manage the game's treasury and configurations. It's responsible for storing the house funds, setting the game parameters (like maximum and minimum stakes), and handling game fees. It also stores the house public key for verifying game outcomes. The module provides functions to adjust game settings, manage the house funds, and ensure the integrity and fairness of the game through cryptographic verification.

The module defines specific error codes to handle exceptional scenarios:

Provide read-only and mutable access to house data properties, enabling operations like querying balance, stake limits, fees, and modifying configurations within authorized contexts:

The plinko::counter_nft module introduces a unique, non-transferable Counter NFT. This Counter object is pivotal for generating a distinct VRF input for each round of the game, thereby ensuring the randomness and fairness of game outcomes. It includes functionalities to create, increment, and destroy these Counter objects. Each Counter is tied to a unique game round and increments

its count with each use, providing a fresh input for the game's randomness mechanism.

Unlike traditional NFTs, the Counter NFT is non-transferable, ensuring it remains tied to its original owner and serves its purpose without the risk of duplication or unauthorized transfer.

Navigate to the setup folder of the Plinko repository and execute the publish.sh script. Refer to the README instructions for deploying the smart contracts and testing them locally.

The Plinko component, a central element of the Plinko game's frontend module, is structured to create an interactive and responsive gaming experience. Written in React, it integrates several features and functions to handle the game's logic and user interactions effectively.

State hooks: The module uses React's useState to manage various game states, including finalPaths , isPlaying , totalWon , and more. These states are vital for tracking the current game status and updating the UI accordingly.

Modal for game initialization: A modal is implemented to facilitate the creation of a new game. It captures the randomness and uses handlePlayClick to initiate the game with the backend.

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

Sequence diagram

Follow the comments in each module's code to understand the logic each creates.

The plinko::plinko module combines various Sui blockchain features, such as coin handling, event emissions, and dynamic object fields, to create on-chain Plinko games.

Error handling is integral to the module, with specific codes indicating various failure states or invalid operations:

Provide read-only access to the game's properties, such as:

Include utilities like:

The plinko::house_data module in the Plinko game is designed to manage the game's treasury and configurations. It's responsible for storing the house funds, setting the game parameters (like maximum and minimum stakes), and handling game fees. It also stores the house public key for verifying game outcomes. The module provides functions to adjust game settings, manage the house funds, and ensure the integrity and fairness of the game through cryptographic verification.

The module defines specific error codes to handle exceptional scenarios:

Provide read-only and mutable access to house data properties, enabling operations like querying balance, stake limits, fees, and modifying configurations within authorized contexts:

The plinko::counter_nft module introduces a unique, non-transferable Counter NFT. This Counter object is pivotal for generating a distinct VRF input for each round of the game, thereby ensuring the randomness and fairness of game outcomes. It includes functionalities to create, increment, and destroy these Counter objects. Each Counter is tied to a unique game round and increments its count with each use, providing a fresh input for the game's randomness mechanism.

Unlike traditional NFTs, the Counter NFT is non-transferable, ensuring it remains tied to its original owner and serves its purpose without the risk of duplication or unauthorized transfer.

Navigate to the setup folder of the Plinko repository and execute the publish.sh script. Refer to the README instructions for deploying the smart contracts and testing them locally.

The Plinko component, a central element of the Plinko game's frontend module, is structured to create an interactive and responsive gaming experience. Written in React, it integrates several features and functions to handle the game's logic and user interactions effectively.

State hooks: The module uses React's useState to manage various game states, including finalPaths , isPlaying , totalWon , and more. These states are vital for tracking the current game status and updating the UI accordingly.

Modal for game initialization: A modal is implemented to facilitate the creation of a new game. It captures the randomness and uses handlePlayClick to initiate the game with the backend.

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

Move modules

Follow the comments in each module's code to understand the logic each creates.

The plinko::plinko module combines various Sui blockchain features, such as coin handling, event emissions, and dynamic object fields, to create on-chain Plinko games.

Error handling is integral to the module, with specific codes indicating various failure states or invalid operations:

Provide read-only access to the game's properties, such as:

Include utilities like:

The plinko::house_data module in the Plinko game is designed to manage the game's treasury and configurations. It's responsible for storing the house funds, setting the game parameters (like maximum and minimum stakes), and handling game fees. It also stores the house public key for verifying game outcomes. The module provides functions to adjust game settings, manage the house funds, and ensure the integrity and fairness of the game through cryptographic verification.

The module defines specific error codes to handle exceptional scenarios:

Provide read-only and mutable access to house data properties, enabling operations like querying balance, stake limits, fees, and modifying configurations within authorized contexts:

The plinko::counter_nft module introduces a unique, non-transferable Counter NFT. This Counter object is pivotal for generating a distinct VRF input for each round of the game, thereby ensuring the randomness and fairness of game outcomes. It includes functionalities to create, increment, and destroy these Counter objects. Each Counter is tied to a unique game round and increments its count with each use, providing a fresh input for the game's randomness mechanism.

Unlike traditional NFTs, the Counter NFT is non-transferable, ensuring it remains tied to its original owner and serves its purpose without the risk of duplication or unauthorized transfer.

Navigate to the setup folder of the Plinko repository and execute the publish.sh script. Refer to the README instructions for deploying the smart contracts and testing them locally.

The Plinko component, a central element of the Plinko game's frontend module, is structured to create an interactive and responsive gaming experience. Written in React, it integrates several features and functions to handle the game's logic and user interactions effectively.

State hooks: The module uses React's useState to manage various game states, including finalPaths , isPlaying , totalWon , and more. These states are vital for tracking the current game status and updating the UI accordingly.

Modal for game initialization: A modal is implemented to facilitate the creation of a new game. It captures the randomness and uses handlePlayClick to initiate the game with the backend.

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

Deployment

Navigate to the setup folder of the Plinko repository and execute the publish.sh script. Refer to the README instructions for deploying the smart contracts and testing them locally.

The Plinko component, a central element of the Plinko game's frontend module, is structured to create an interactive and responsive gaming experience. Written in React, it integrates several features and functions to handle the game's logic and user interactions effectively.

State hooks: The module uses React's useState to manage various game states, including finalPaths , isPlaying , totalWon , and more. These states are vital for tracking the current game status and updating the UI accordingly.

Modal for game initialization: A modal is implemented to facilitate the creation of a new game. It captures the randomness and uses handlePlayClick to initiate the game with the backend.

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and

source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

Frontend

The Plinko component, a central element of the Plinko game's frontend module, is structured to create an interactive and responsive gaming experience. Written in React, it integrates several features and functions to handle the game's logic and user interactions effectively.

State hooks: The module uses React's useState to manage various game states, including finalPaths , isPlaying , totalWon , and more. These states are vital for tracking the current game status and updating the UI accordingly.

Modal for game initialization: A modal is implemented to facilitate the creation of a new game. It captures the randomness and uses handlePlayClick to initiate the game with the backend.

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

State management and setup

State hooks: The module uses React's useState to manage various game states, including finalPaths , isPlaying , totalWon , and more. These states are vital for tracking the current game status and updating the UI accordingly.

Modal for game initialization: A modal is implemented to facilitate the creation of a new game. It captures the randomness and uses

handlePlayClick to initiate the game with the backend.

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

UI components and styling

The MatterSim and PlinkoSettings components are the foundation for the Plinko frontend. To see code for all components and source files, see the Plinko repo .

The MatterSim component plays a pivotal role in the Plinko game by rendering the game board with a realistic physical representation of the dropping Plinko balls. It leverages the robust capabilities of Matter.js , a renowned physics engine, to simulate the intricate dynamics of the Plinko game's environment.

The randomness integral to the game's fairness and unpredictability, as detailed in the counter_nft module , dictates the path of each ball dropped by the player. To align the Plinko balls' movement with the predetermined paths derived from on-chain data, MatterSim applies physics principles, such as gravity, to ensure natural ball descent. Additionally, to accurately guide the balls along these specific trajectories, MatterSim introduces subtle, custom forces. These adjustments are calibrated to not only ensure compliance with the paths determined by the game's underlying blockchain mechanics but also to maintain a visually coherent and physically plausible ball movement, to better simulate realism.

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

Plinko settings component

The PlinkoSettings component is an integral part of the user interface in the Plinko game, enabling players to customize their gameplay experience according to their preferences. This React component allows users to select the number of Plinko balls they want to drop, set the bet size for each ball, and initiate the game round by pressing the Play button.

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

User interaction and feedback

Game Initiation: After selecting the desired number of balls and setting the bet size for each, players can initiate a new game by pressing the Play button. This action starts the game, with the button becoming disabled during gameplay to prevent new games from being initiated until the current game concludes and the last ball has reached the end. Links are also provided for players to view game details on a Sui network explorer for transparency and engagement.

Related links