Hexxagon on Cardano: Game Design

Table of Contents

Smart Contracts

Smart Contract	InitialiseGameSC	
Value	Bet Amount	It can be ADA, native tokens, or any combination. This is the amount player1 deposits to create a game.
Datum	Player A, Turn Duration, Board	Player A creates a game with those settings and waits for someone to join. - Player A registers his identifier (NFT) for authentication purposes when canceling or playing the game. - Sets the Turn duration limit - and sets the Board
Redeemer	Add Player or Withdraw	Player B will Initialize the game using 'Add Player' as a redeemer. If no one joins the game, Player A can cancel the game and claim back his deposit, using 'Withdraw' as a redeemer.
Purpose	The purpose of this Smart contract is to Initialize the game. Player A creates a game by sending a UTxO (with the Bet Amount as Value and Game Settings as Datum) to this smart contract. Then, Player B will join and start the game by consuming the UTxO and sending a new UTxO to the RunGameSC to run the game. The Smart contract will make sure: 1) Player B deposits the same amount as Player A. 2) The correct datum at the output UTxO (@RunGameSC): a) Players are Player A (from datum) + Player B (from redeemer) b) [1]Turn duration not modified c) [2]Player Turn d) [3]State of the game = State 0 (Not Modified) Or Player A cancels the game by consuming the UTxO and returning the value to his wallet. The Smart contract will make sure: 1) The correct user (Player A) is canceling the game by checking the presence of the registered NFT in the input UTxOs.	

[1] Maybe we should also check that it's >= to some minimum requirement.

[2] Should there be randomness here? Is there any advantage for the starting player? If yes then the initialisation process will slightly change because we would be adding a commit/reveal step.

[3] Here we can add extra checks: for example the game is not Game over already, or max initial filled hexagons, etc.. And if we want max security (which I don't know if necessary) then we should hard code the board in the smart contract code either by replacing the Board in the datum by a BoardType, and then depending on the BoardType we will check for state 0, or making the smart contract parametrized by a Board. We can start by excluding these checks, and adding them later on if needed.

Smart Contract	**RunGameSC**	
Value	Total Bet Amount	
Datum	Players, Turn duration, State of the Game	Game Info: - Players and Turn duration are fixed throughout the game (Read only) - State of the Game changes every move (Read & Write)
Redeemer	Play Turn, Game Over, Timeout	The winning player is wrapped by GameOver (for external use. It is an easy way to expose the winner so that other smart contracts can easily verify a winner by accessing '`txInfoRedeemers`' in TxInfo.)
Purpose	The purpose of this smart contract is to: - **PlayTurn** 1) Make sure the Total Bet Amount remains intact 2) Players and Turn duration stays the same 3) The state of the game is updated legally. a) Current time before the deadline b) Player's turn c) The board move is legal - [4]**GameOver Player** 1) Make sure it is Game Over. 2) Check if the winner == Player 3) Check if the Total Bet Amount is going to the winner - **Timeout** 1) Ensure the current time is after the deadline. 2) Check that the total bet amount is going to the opponent player (i.e., Not the current player)	

[4] Should there be draw case?

Score Tracking System

This part tracks the player's info (score, etc…). We will make use of CIP-68. So, we will create a pair of NFTs for each player:

1) Player's NFT: held by the player (This NFT will also be used to play the game)
2) Reference NFT: held by a smart contract (holds the player's info (ex. score) in the UTxO's datum)

So, we need a minting policy to create these NFTs and a smart contract that holds and manages the reference NFT.

This part is separate from the game itself; multiple games can use it. In other words, it can be part of a large gaming platform. So it's a good exercise if we build it for our game. Plus, Player's NFT can have multiple use cases other than just for player identification and score tracking. For example, it can function as a ticket to be purchased to play the game/s, can grant you access to be part of a DAO, or can also have an expiration date that needs to be renewed by paying a subscription fee, etc…

Minting Policy	PlayerIdentifierMP	
Redeemer	CreatePlayer or DeletePlayer	
Purpose		

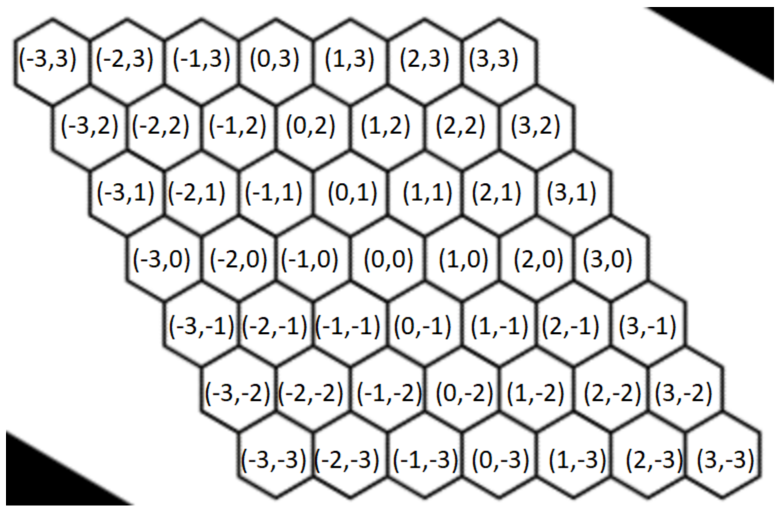
Smart Contract	RefNFTManagerSC	
Value	minADA + RefNFT	
Datum	`[metadata, version, extra]`	
Redeemer	Update or Burn	
Purpose	Track the Player's info (Score)	

Custom Data Types

- data Player = RedPlayer CurrencySymbol TokenName | BluePlayer CurrencySymbol TokenName
- data Hexagon = Empty | Red | Blue
- type Position = (Int, Int)
- type Board = Map Position Hexagon
- data Initialization = Add Player | Withdraw
- data GameSettings = Settings
  ```
  { getPlayer         :: Player
  , getTurnDuration   :: PosixTime
  , getBoardS0        :: Board
  }
  ```
- data GameState = Game
  ```
  { getPlayer'sTurn   :: Player
  , getDeadline       :: PosixTime
  , getBoard          :: Board
  }
  ```
- data GameInfo = GameInfo
  ```
  { getPlayers        :: [Player]
  , getTurnDuration   :: PosixTime
  , getGameState      :: GameState
  }
  ```
- data RunGame = PlayTurn | GameOver Player | TimeOut

- Note: If a Hexagon is Blank, it won't have a key (Position) in the Map (Board). This way, we can create custom Boards by setting the keys (Positions) of the Map (Board). And the smart contract, in one way or another, would make sure no new keys are inserted or deleted.

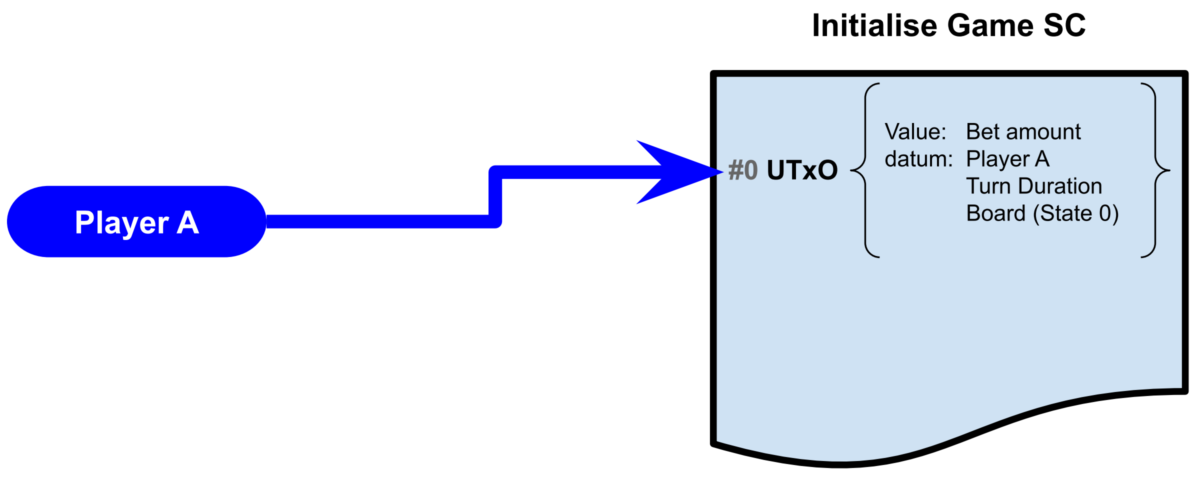
Calculations



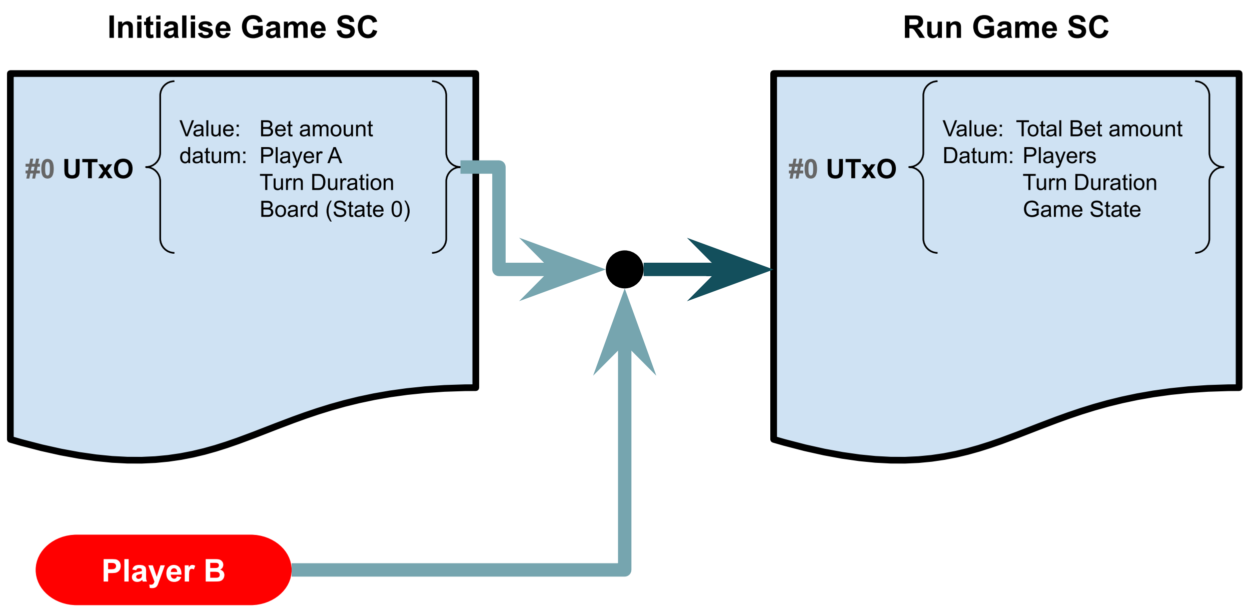
$$distance = \frac{|\Delta x| + |\Delta y| + |\Delta x - \Delta y|}{2}$$

Flow Charts

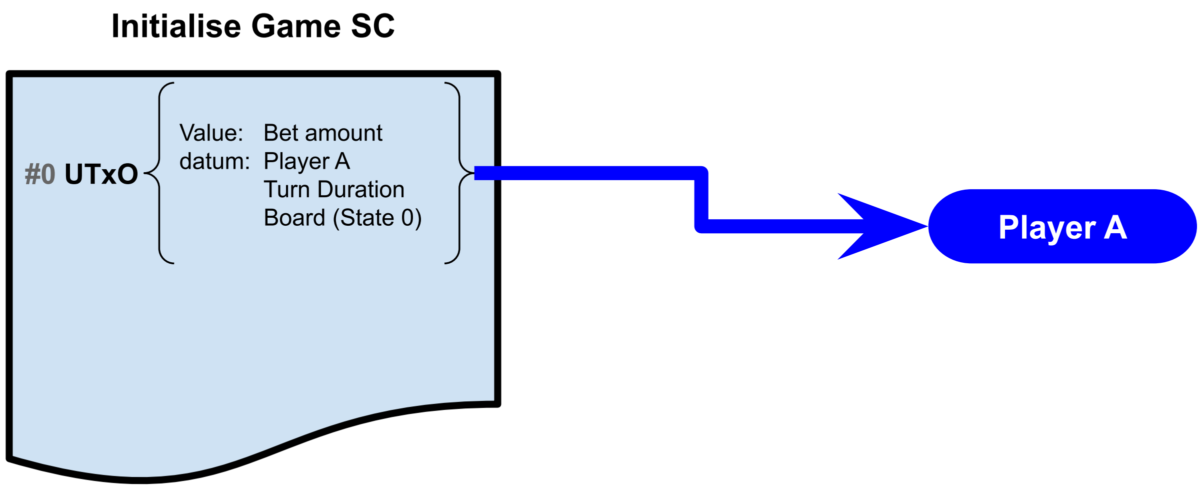
1 - Player A creates a game



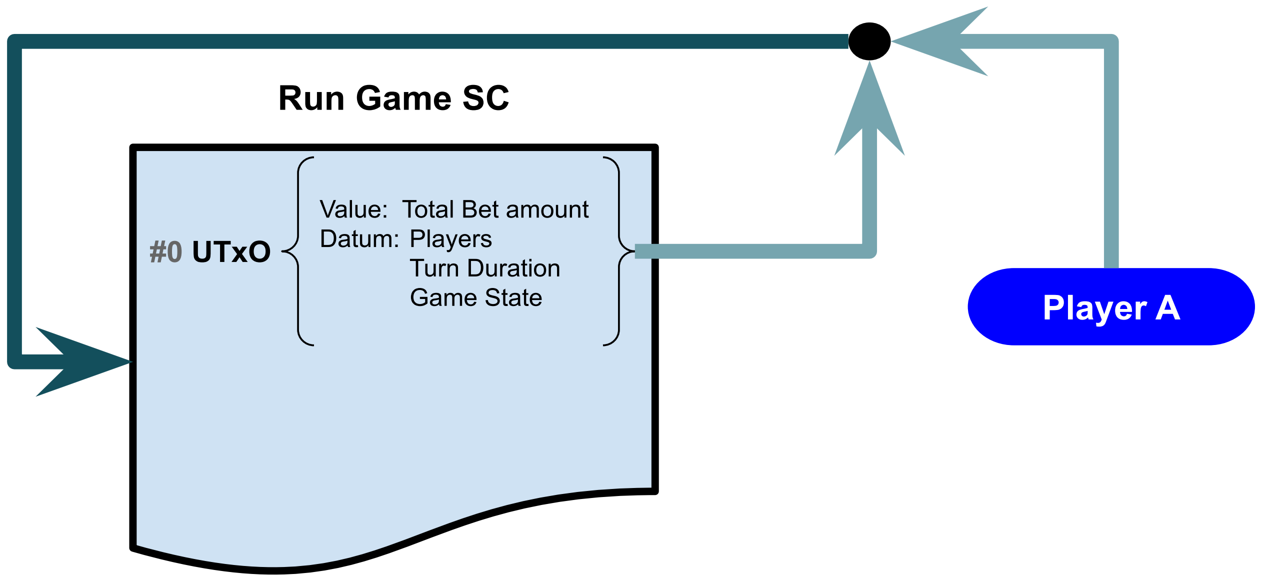
2A - Player B joins the game

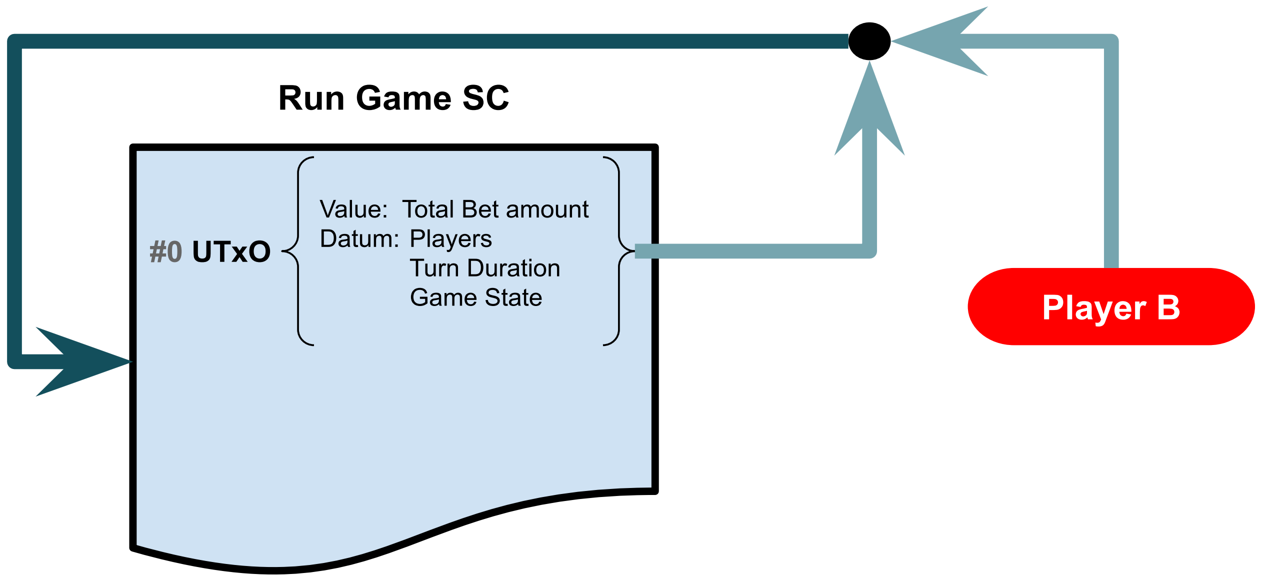


2B - Player A cancels the game

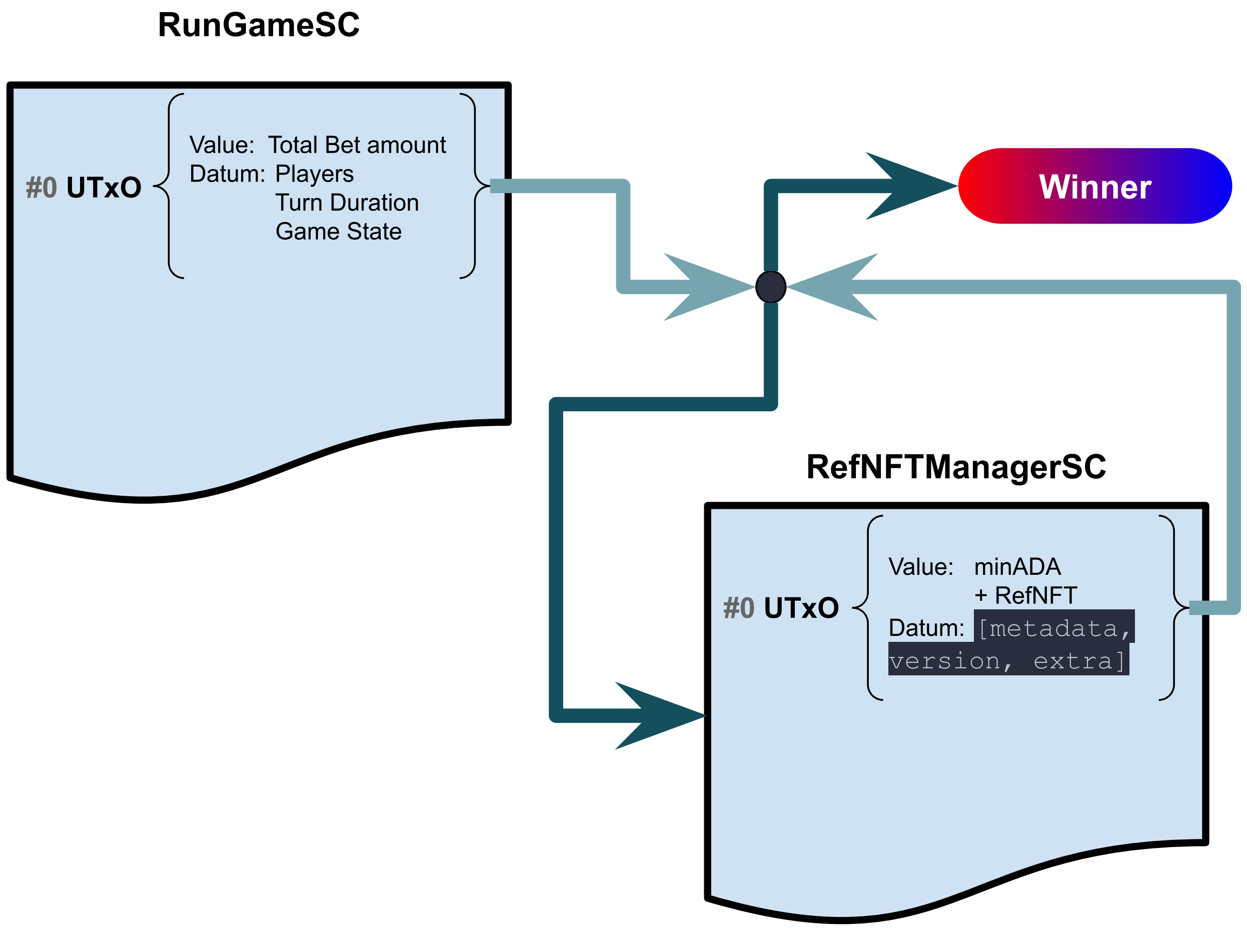


3 - Running the game





4 - Winner claims the reward and updates his score


RunGameSC
#0 UTxO
Value: Total Bet amount
Datum: Players
Turn Duration
Game State
Winner
RefNFTManagerSC
#0 UTxO
Value: minADA
+ RefNFT
Datum: [metadata, version, extra]