Ribbit Royale

Baron Baker, Chase Bennett, Adam Bobich, Aidan Caughey, Ryan Dobkin, Luke Garci

Problems & Motivations

- Lack of free to play party games on the market
- Introduction of unique mechanics and physics in minigames
- LAN Multiplayer revival
- Create a casual yet competitive game for everyone to enjoy

Toolset

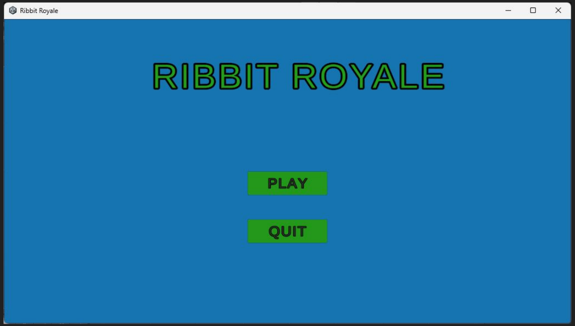
- Unity game engine and C#
- Netcode for GameObjects

  - Handles multiplayer synchronization
- Unity Relay

  - Networking solution that connects players in a peer-to-peer fashion using a server-client environment.
- Unity Lobby

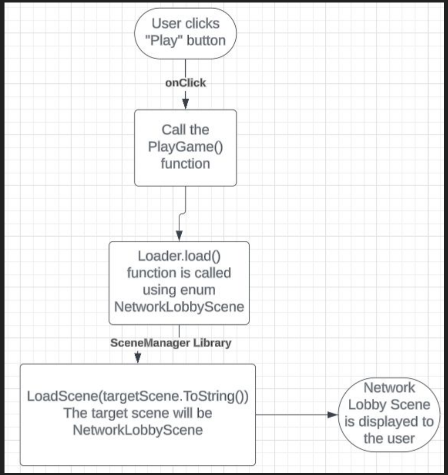
  - A way for players to discover and connect to each other
- Aseprite

Main Menu Scene



- Main menu scene
- Play button
- Quit button

MainMenuUI.cs



User clicks "Play" button

onClick

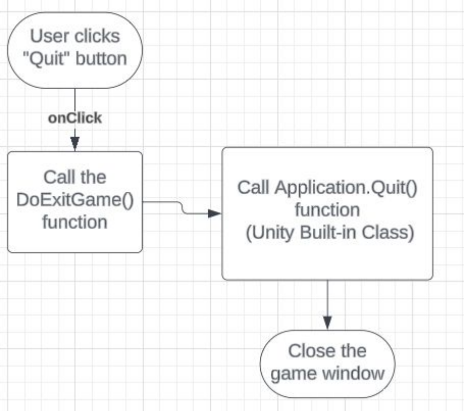
Call the PlayGame() function

Loader.load() function is called using enum NetworkLobbyScene

SceneManager Library

LoadScene(targetScene.ToString()) The target scene will be NetworkLobbyScene

Network Lobby Scene is displayed to the user



User clicks "Quit" button

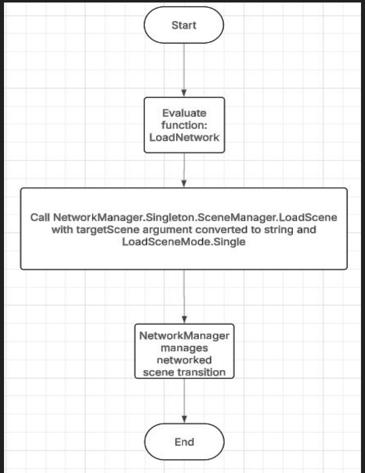
onClick

Call the DoExitGame() function

Call Application.Quit() function (Unity Built-in Class)

Close the game window

Loader.cs



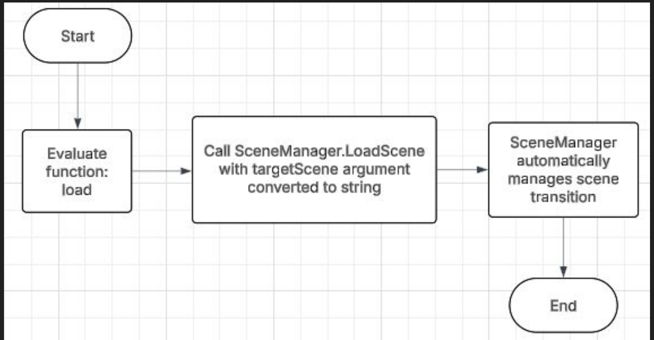
Start

Evaluate function: LoadNetwork

Call NetworkManager.Singleton.SceneManager.LoadScene with targetScene argument converted to string and LoadSceneMode.Single

NetworkManager manages networked scene transition

End



Start

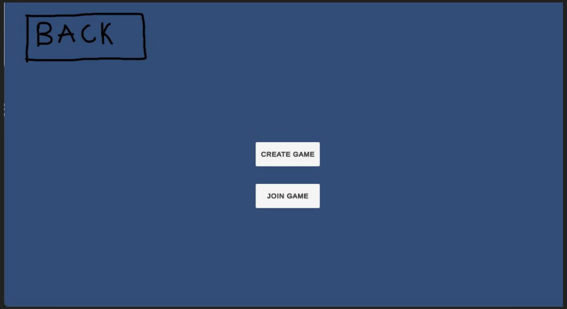
Evaluate function: load

Call SceneManager.LoadScene with targetScene argument converted to string

SceneManager automatically manages scene transition

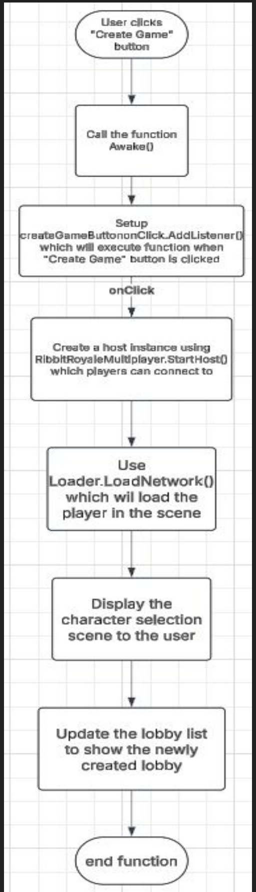
End

Network Lobby Scene



- Create Game button
- Join Game button
- Back button

LobbyUI.cs



User clicks "Create Game" button

Call the function Awake()

Setup createGameButtononClick.AddListener() which will execute function when "Create Game" button is clicked

onClick

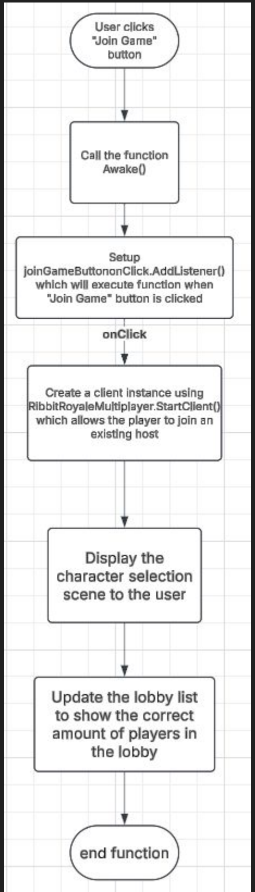
Create a host instance using RibbitRoyaleMultiplayer.StartHost() which players can connect to

Use Loader.LoadNetwork() which will load the player in the scene

Display the character selection scene to the user

Update the lobby list to show the newly created lobby

end function



User clicks "Join Game" button

Call the function Awake()

Setup joinGameButtononClick.AddListener() which will execute function when "Join Game" button is clicked

onClick

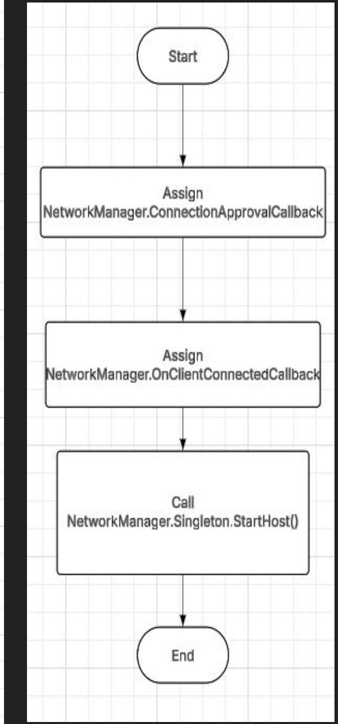
Create a client instance using RibbitRoyaleMultiplayer.StartClient() which allows the player to join an existing host

Display the character selection scene to the user

Update the lobby list to show the correct amount of players in the lobby

end function

RibbitRoyaleMultiplayer.cs



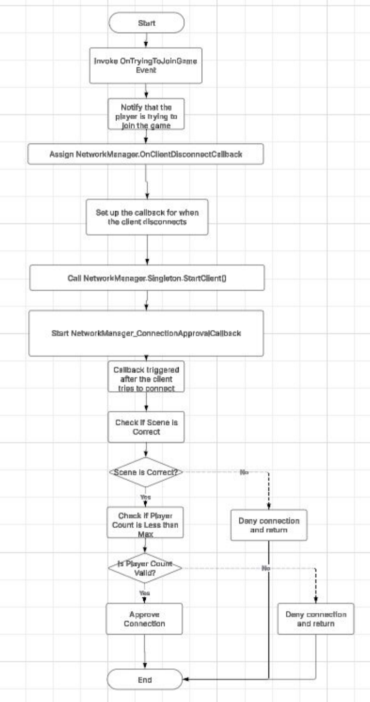
Start

Assign NetworkManager.ConnectionApprovalCallback

Assign NetworkManager.OnClientConnectedCallback

Call NetworkManager.Singleton.StartHost()

End



Start

Invoke OnTryingToJoinGame Event

Notify that the player is trying to join the game

Assign NetworkManager.OnClientDisconnectCallback

Set up the callback for when the client disconnects

Call NetworkManager.Singleton.StartClient()

Start NetworkMansger_ConnectionApprovalCallback

Callback triggered after the client tries to connect

Check If Scene is Correct

Scene is Correct?

No

Yes

Check if Player Count is Less than Max

Deny connection and return

Is Player Count Valid?

No

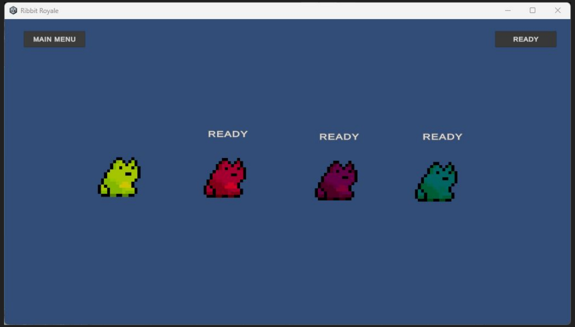
Yes

Approve Connection

Deny connection and return

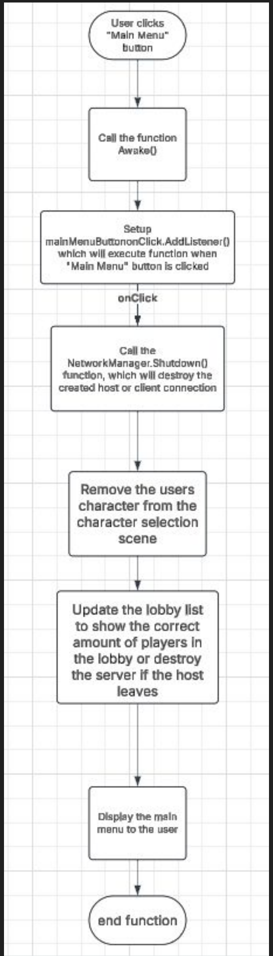
End

Character Selection Scene



- Display user as a frog in the lobby
- Ready button
- Main Menu button

CharacterSelectionUI.cs



User clicks "Main Menu" button

Call the function Awake()

Setup mainMenuButtononClick.AddListener() which will execute function when "Main Menu" button is clicked

onClick

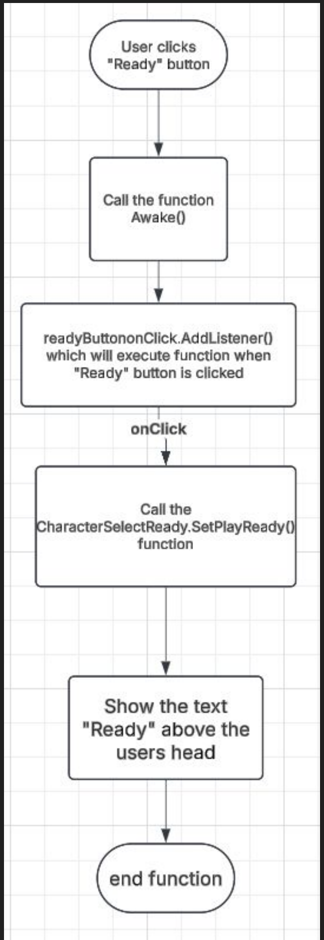
Call the NetworkManager.Shutdown() function, which will destroy the created host or client connection

Remove the users character from the character selection scene

Update the lobby list to show the correct amount of players in the lobby or destroy the server if the host leaves

Display the main menu to the user

end function



User clicks "Ready" button

Call the function Awake()

readyButtononClick.AddListener() which will execute function when "Ready" button is clicked

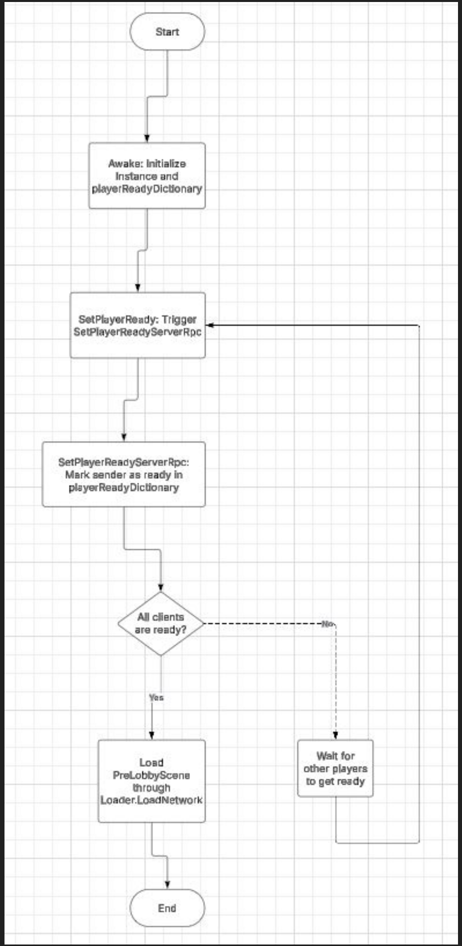
onClick

Call the CharacterSelectReady.SetPlayReady() function

Show the text "Ready" above the users head

end function

CharacterSelectReady.cs



Start

Awake: Initialize Instance and playerReadyDictionary

SetPlayerReady: Trigger SetPlayerReadyServerRpc

SetPlayerReadyServerRpc: Mark sender as ready in playerReadyDictionary

All clients are ready?

No

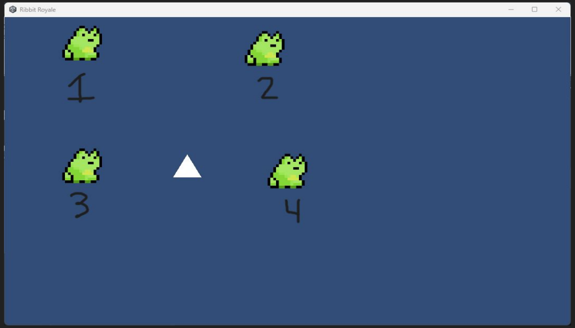
Yes

Load PreLobbyScene through Loader.LoadNetwork

Wait for other players to get ready

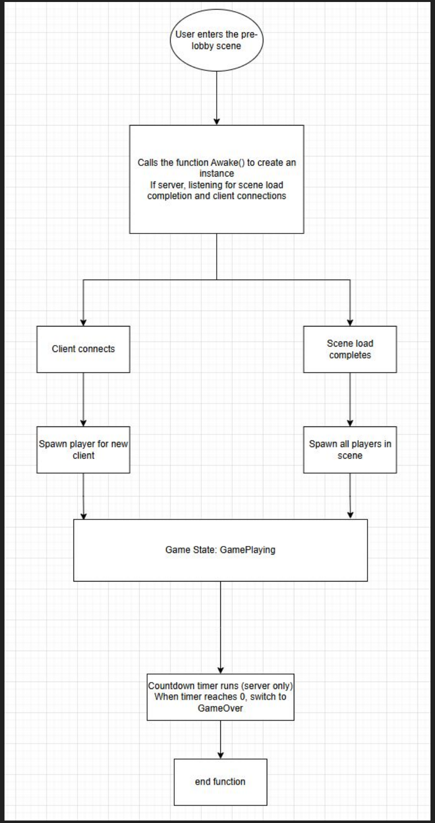
End

Pre-lobby Scene



- Manages the Game State
- Handles Player Connections

PreLobbyManager.cs



User enters the pre-lobby scene

Calls the function Awake() to create an instance
If server, listening for scene load completion and client connections

Client connects

Spawn player for new client

Scene load completes

Spawn all players in scene

Game State: GamePlaying

Countdown timer runs (server only)
When timer reaches 0, switch to GameOver

end function

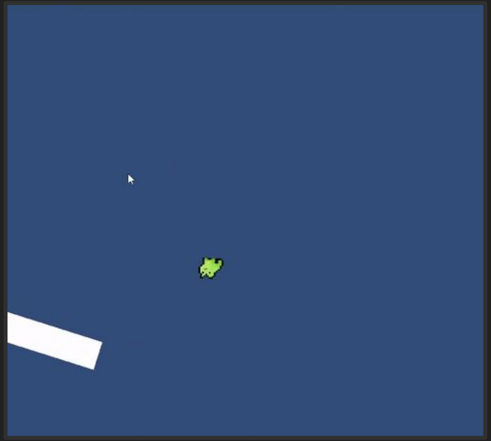
Minigames

- Lickity Split
- Swamp Spotting
- Snake Escape

Lickity Split

Use your tongue to swing to the finish line faster than your opponents

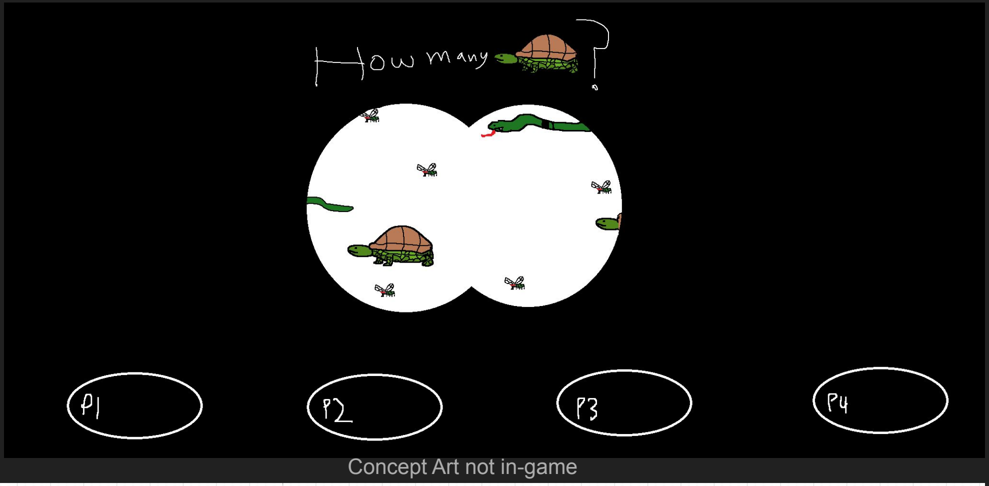
- Movement
  - A / D control horizontal movement
  - If the mouse is clicked, create a tongue object and attach to the nearest node if in range.
  - If mouse released, destroy the tongue connection.
  - If a frog lands, change it's velocity so it bounces off.
- Swinging
  - Use a Unity Spring Joint to connect the frog to the node (swinging point).
  - Add a small boost of speed in the direction of the swing, giving more satisfying launches.
- Goal - Reach the end platform
  - Use the momentum from your swings to reach the platform at the top of the course
  - Upon reaching the platform, your time will be recorded and compared to the other players



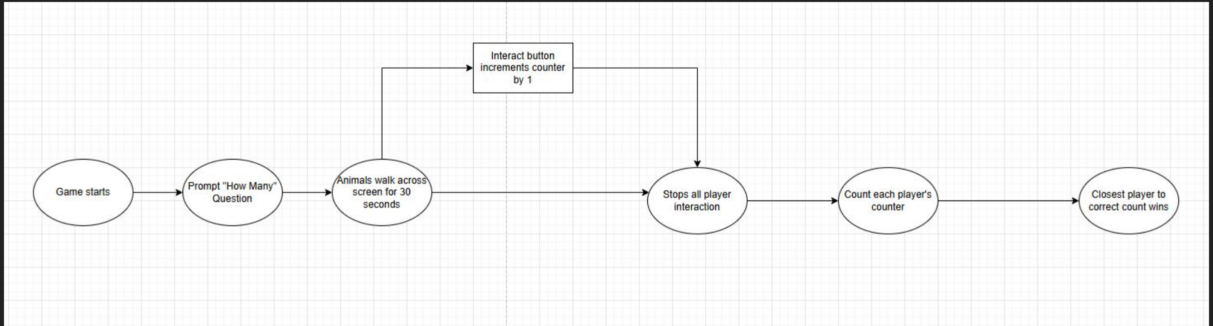
Swamp Spotting

Count the creatures of the swamp

- Objective is to count the specific animal
- Animals walk across screen
    - Players press interact button to count
- Award points based on who's the closest



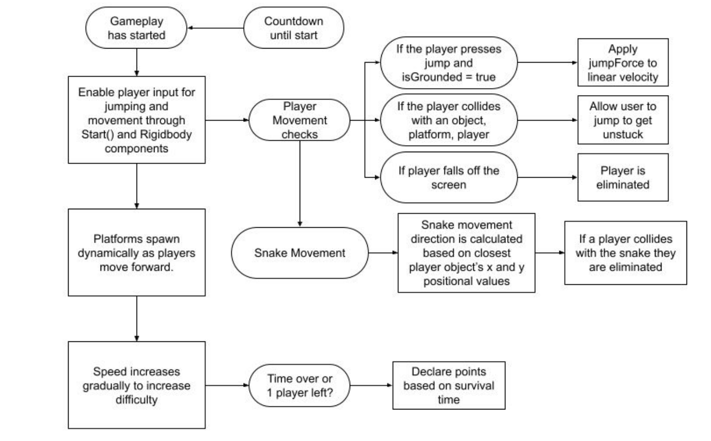
Concept Art not in-game


Game starts
Prompt "How Many" Question
Animals walk across screen for 30 seconds
Interact button increments counter by 1
Stops all player interaction
Count each player's counter
Closest player to correct count wins

Snake Escape

Hop for your life, every frog for themselves

- Escape from the snake until you reach the time runs out or you reach the finish line!
- Players can control the frog to move left, right, and up.
- Players will have to jump from platform to platform, across trees or puddles, etc.
- Failure to stay alive until time runs out or falling out of the stage will result in a loss.
- Points are awarded depending on placement among players.



Countdown until start

Gameplay has started

Enable player input for jumping and movement through Start() and Rigidbody components

Player Movement checks

If the player presses jump and isGrounded = true

Apply jumpForce to linear velocity

If the player collides with an object, platform, player

Allow user to jump to get unstuck

If player falls off the screen

Player is eliminated

Platforms spawn dynamically as players move forward.

Snake Movement

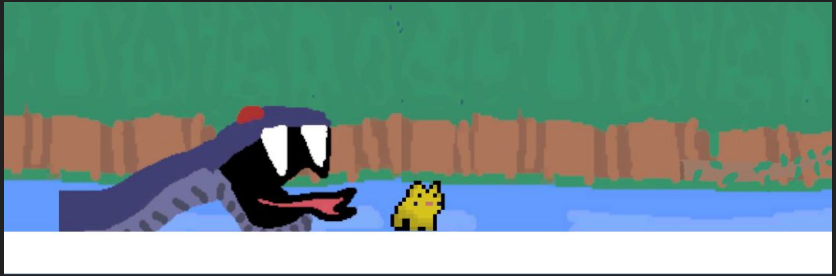
Snake movement direction is calculated based on closest player object's x and y positional values

If a player collides with the snake they are eliminated

Speed increases gradually to increase difficulty

Time over or 1 player left?

Declare points based on survival time



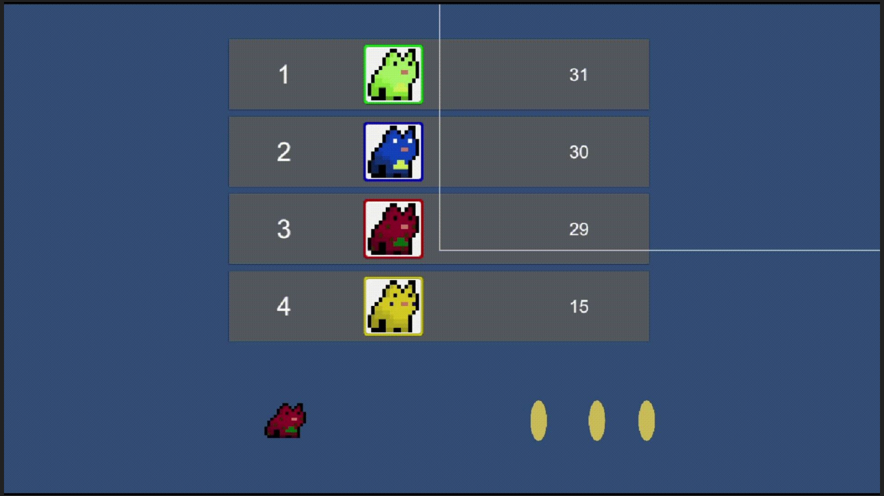
FrogController.cs and SnakeController.cs

Future functionality of SnakeChaseScore.cs included

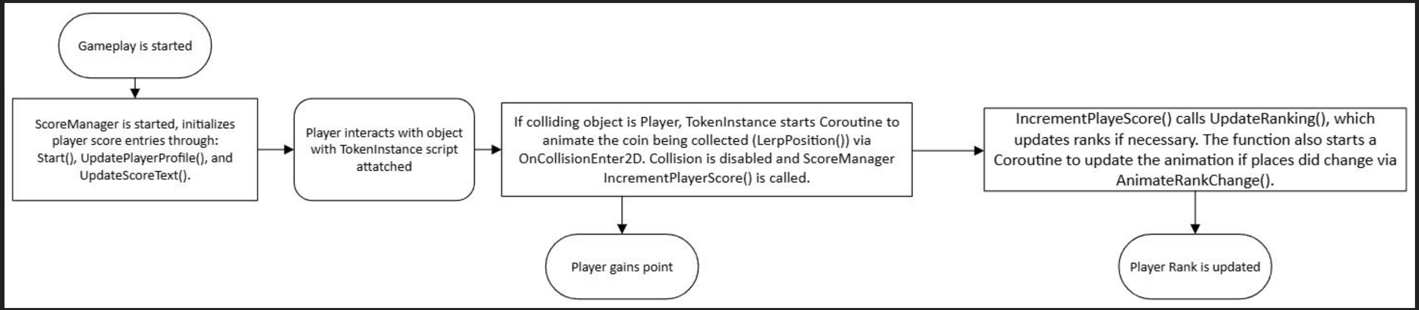
Sprites and animations are still WIP

Scoring

- Displayed to the user as an overlay
- Populates when players interact
- Lerp animation on place change
- Updates scores and rankings
- Avatars and Colors assigned assigned
- Can be toggled with button press (Tab)



1	🟩	31
2	🟦	30
3	🟥	29
4	🟨	15



Gameplay is started

ScoreManager is started, initializes player score entries through: Start(), UpdatePlayerProfile(), and UpdateScoreText().

Player interacts with object with TokenInstance script attatched

If colliding object is Player, TokenInstance starts Coroutine to animate the coin being collected (LerpPosition()) via OnCollisionEnter2D. Collision is disabled and ScoreManager IncrementPlayerScore() is called.

Player gains point

IncrementPlayeScore() calls UpdateRanking(), which updates ranks if necessary. The function also starts a Coroutine to update the animation if places did change via AnimateRankChange().

Player Rank is updated

Testing

- Unit Testing (Input Handling, Physics, Game Logic)
- Network Testing (LAN, Latency handling, Synchronization)
- UI Testing (Scoreboard, HUD)
- Performance Testing (Max Players, Network Load, FPS stability)
- Playtesting between group members and outside participants

Documentation & Coding Guidelines

- In game instructions
  - Before each minigame begins, inform the players of the game controls and win conditions
  - Any in-game components that don’t include explicit instructions should have intuitive behavior, such as the play button being labeled “Play”

- Scripts and other Components will be labeled and sorted by their use
  - A script for the player controller in a Tongue Swinging game would not be called **PlayerController.cs**, but instead **TSPlayerController.cs**
- Scripts will be modularized as much as possible, making it easy for new programmers to know what’s going on even without extensive comments

Thank You!