Project Football
Game Design Document

Game Overview

1. **Clone game of "Football Head"**
2. **Side-view 2D physics football game.**
3. **Online 2-player Multiplayer game**

Game Loop

a. **One player on each side of the field with goals behind.**
b. **Time limit 1-minute.**
c. **Players move left and right, jump, and kick or head the ball toward the goal.**
d. **The ball reacts with basic physics, so timing your jumps and shots matters.**
e. **Matches are short and fast-paced, so reaction speed and positioning are key.**

Specifications

Network:

Host-client (Client hosted Multiplayer) system.
No dedicated server.
Host - calculates all physics and broadcasts "gamestate"
Client - only sends inputs (applies received gamestate).
60hz tickrate (sufficient for 1 minute game)

Library: SFML-3.0.2 (For graphics, network, audio and system)
Physics:

Must Choose:
Option 1: Start from scratch. Option 2: Box2D library

Example of GameState

```
struct GameState {
   sf::Vector2f player1Pos;
   sf::Vector2f player2Pos;
   sf::Vector2f ballPos;
   sf::Vector2f ballVelocity;
   int score1, score2;
   float timeRemaining;

   // SFML Packet Serialization/Deserialization (for sequential stream of bytes)
   sf::Packet& serialize(sf::Packet& packet);
   void deserialize(sf::Packet& packet);
};
```

Class Responsibility

GameScene (Host Mode)
├─ receives Input from both players
├─ updates Physics (Player, Ball, Collisions)
├─ sends GameState to Client
└─ renders local view

GameScene (Client Mode)
├─ sends Input only
├─ receives GameState
├─ interpolates for smooth rendering
└─ renders remote view

Communication Protocol

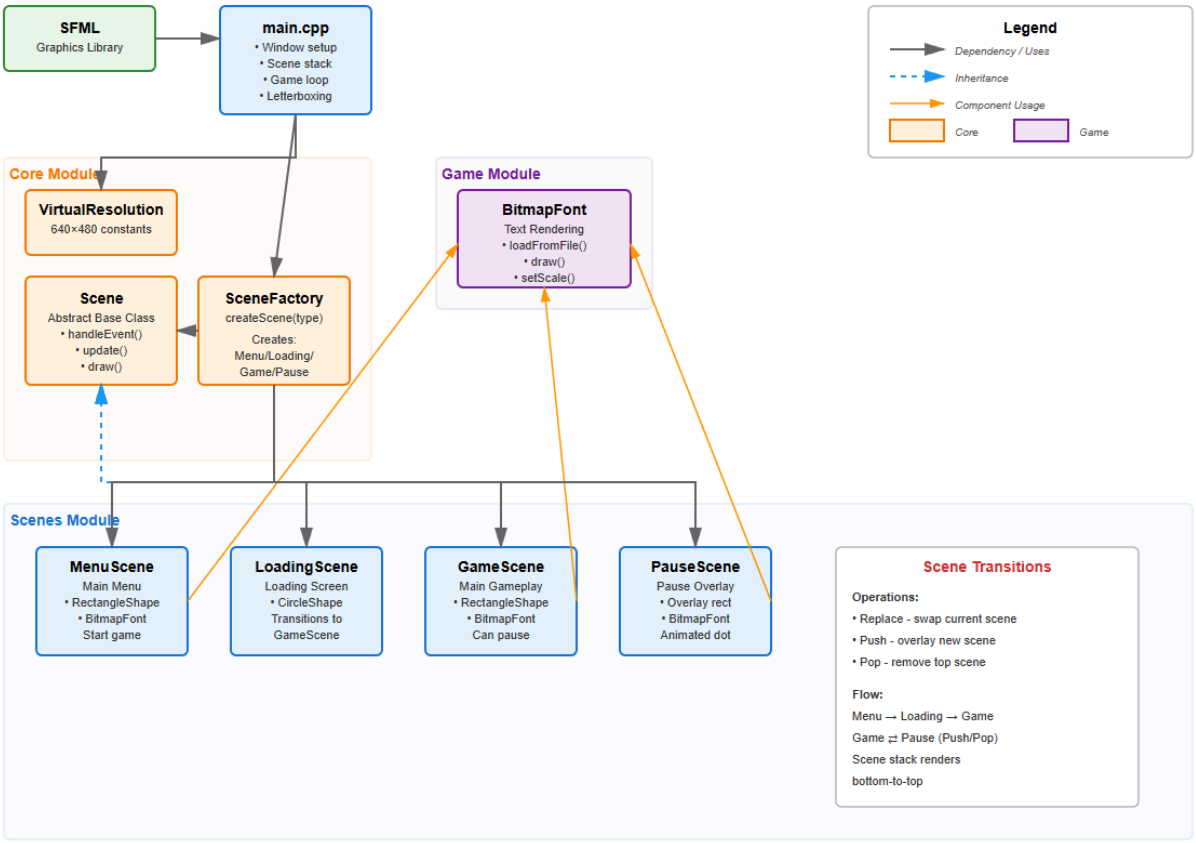
Packet Types:
1. InputPacket (Client → Host)
- uint8: keys pressed (bitfield: A|D|W|L)
- uint32: sequence number

2. StatePacket (Host → Client)
- GameState struct (serialized)
- uint32: sequence number

Test Scenarios:

a. **Host quits mid-game**
b. **Client disconnects**
c. **Both players spam kick simultaneously**
d. **Ball goes out of bounds (just as fences like the original game)**
e. **Timer hits 0:00 with tie score (just end game as draw?)**

Current Base Framework Structure



Development Timeline:

Phase 1:

a. Player movements (move, jump, kick)
b. Ball physics (gravity + bouncing)
c. Goal detection + scoring (resetting game positions when scored)
d. 1 minute timer

Phase 2:

a. Lobby Scene (simple UI for entering IP as client, or choose host)
b. Network manager implementation
c. Host: standby for connection from client
d. Client: connect to host

Phase 3:

a. Sending of input packets from client
b. Synchronization with “gamestate”
c. Basic interpolation of movements for smoothening

Phase 4:

a. **Result Scene (show winner and score)**
b. **Rematch button**
c. **Polishing of animations and art**

Aspects to be Ignored:

a. **Cheat-prevention**
b. **Data encryption in packets**
c. **Reconnection logic on connection lost**

Controls

A - left
D - right
W - space
L - kick

(default for now)

Screenshot