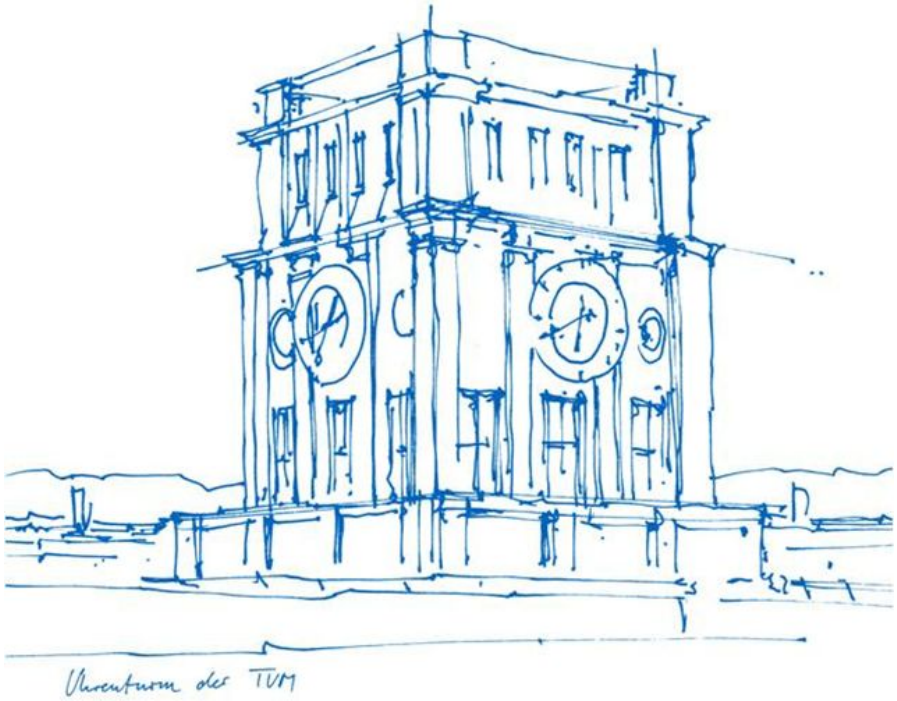
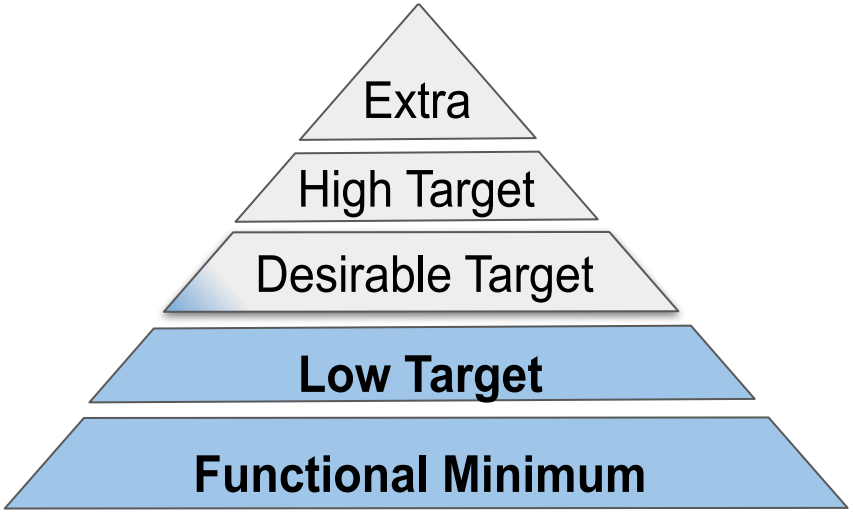
Interim Demo
Rebomb

Yaxuan Dai
Mahdis Sabzevarzadeh
Miguel Trasobares
Jialin Yang



Technical University of Munich
TUM School of Computation, Information and Technology
Chair of Computer Graphics and Visualization
Garching 04.12.2024

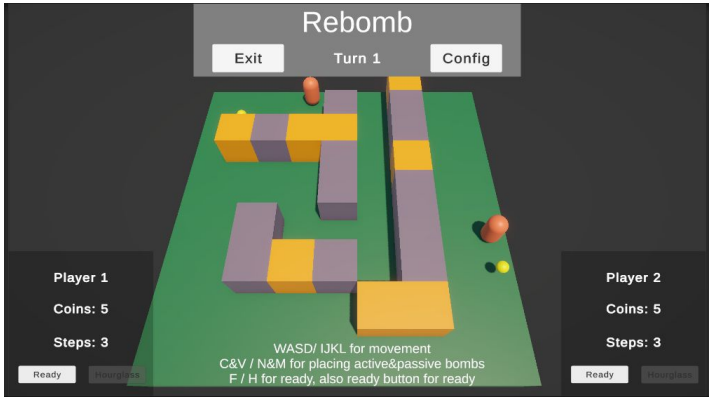
1 Task Progression

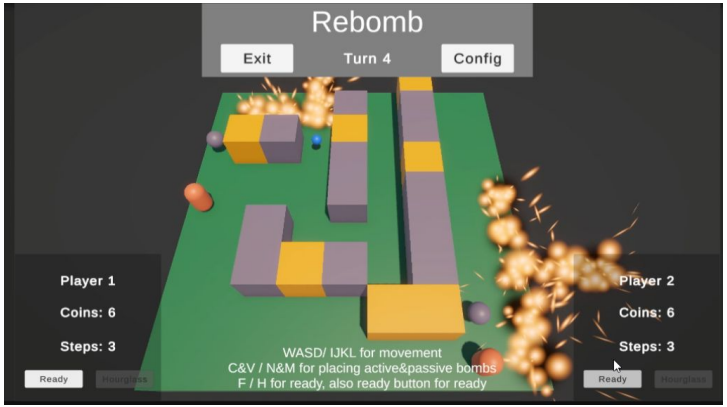


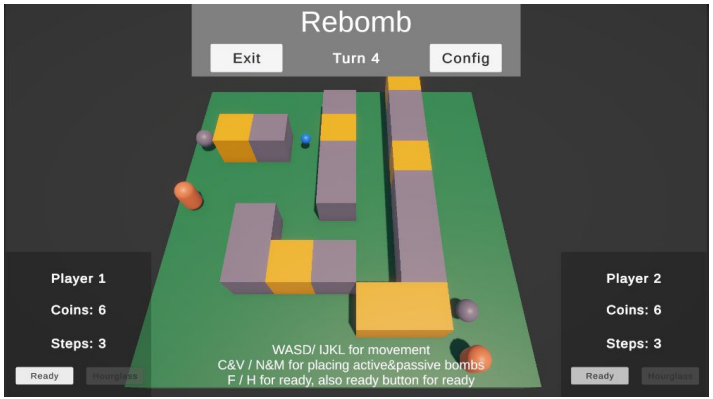
Timeline Update

Date	Milestone	Week	Layer	Task	Time		Owner	State
					Expect	Actual		
Nov 06-12	**Prototype**	1	Prototype	physical Prototype	4 * 5	4 * 5	All	DONE
			Minimum	simple assets	5	5	All	DONE
				simple map & static object	5	3	Jialin	DONE
				player move & item place	5	1	Jialin	DONE
				simple GUI	5	2	Mahdis	DONE
Nov 13-19		2		active & passive bomb	6	7	Miguel	DONE
				resourse system	6	1	Yaxuan	DONE
				interactive map & object	6	6	All	DONE
				turn-based gameplay	6	10	Yaxuan	DONE
				version integration	4 * 4	4 * 2	All	DONE
Nov 20-26		3	Low	specifical assets	8	2	Miguel	TODO
				time travel mechanism 1/2	8	6	Jialin	DONE
				weapon & interactive object	8	8	Mahdis	DONE
				full GUI	8	8	Yaxuan	DONE
				version integration	4 * 2	4 * 2	All	DONE
Nov 27-Dec 03	**Interim demo**	4		time travel mechanism 2/2	8	10	Jialin	DONE
				cascaded explosion refine	8	2	Miguel	TODO
			Desirable	map generation 1/3	8			
				local multiplayer 1/3	8	10	Yaxuan	DONE
				version integration	4 * 2	4 * 2	All	TODO
Dec 04-10		5		map generation 2/3	8			
				local multiplayer 2/3	8		Yaxuan	DONE
				explosion effects 1/2	8			
				more weapon and objects	8			
				version integration	4 * 2		All	
			Desirable	map generation 3/3	6			
				local multiplayer 3/3	6		Yaxuan	DONE

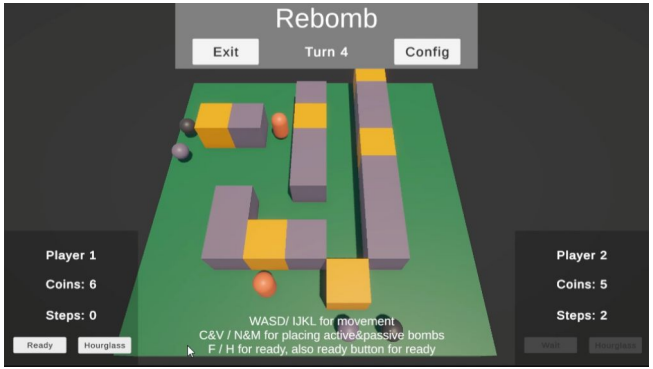
Current Status



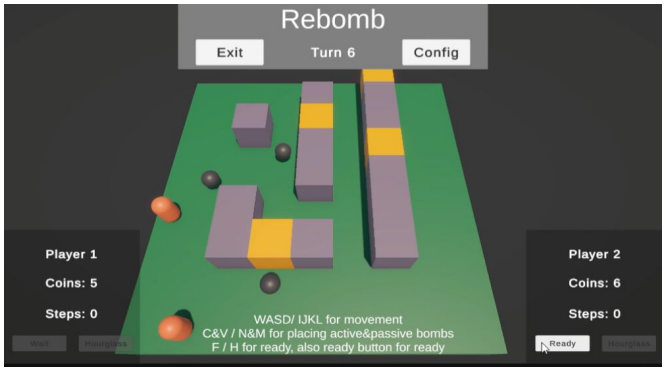




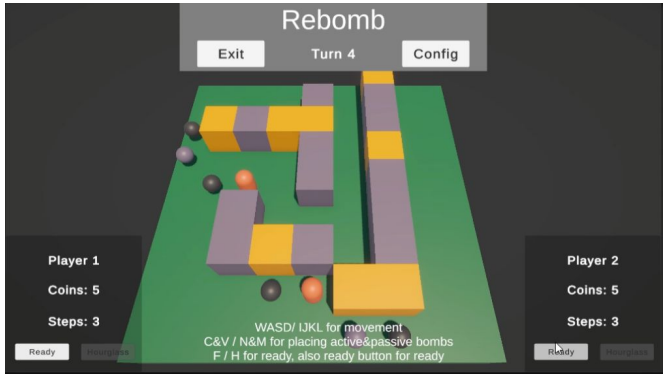
Time Travel



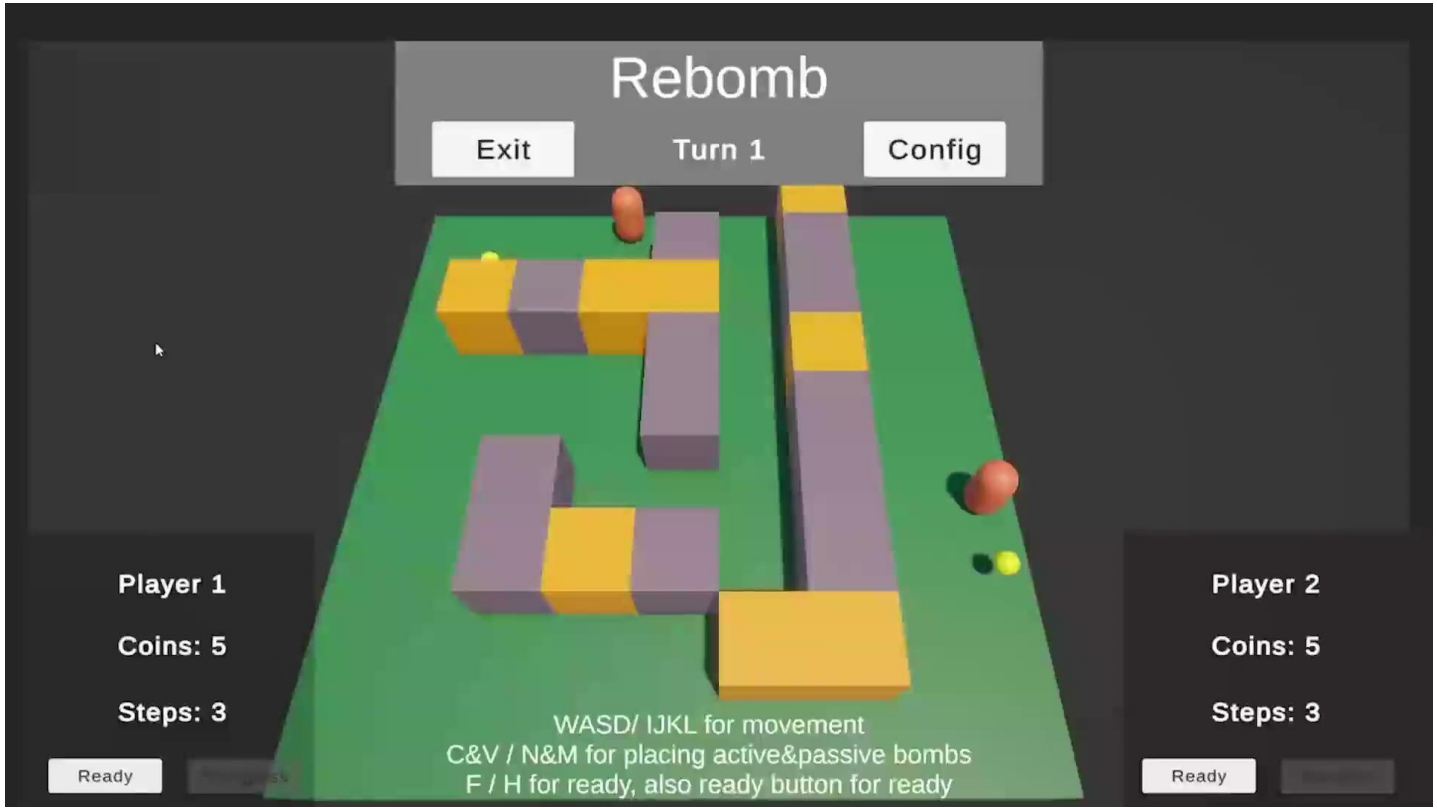
State of Turn 4



After hourglass usage



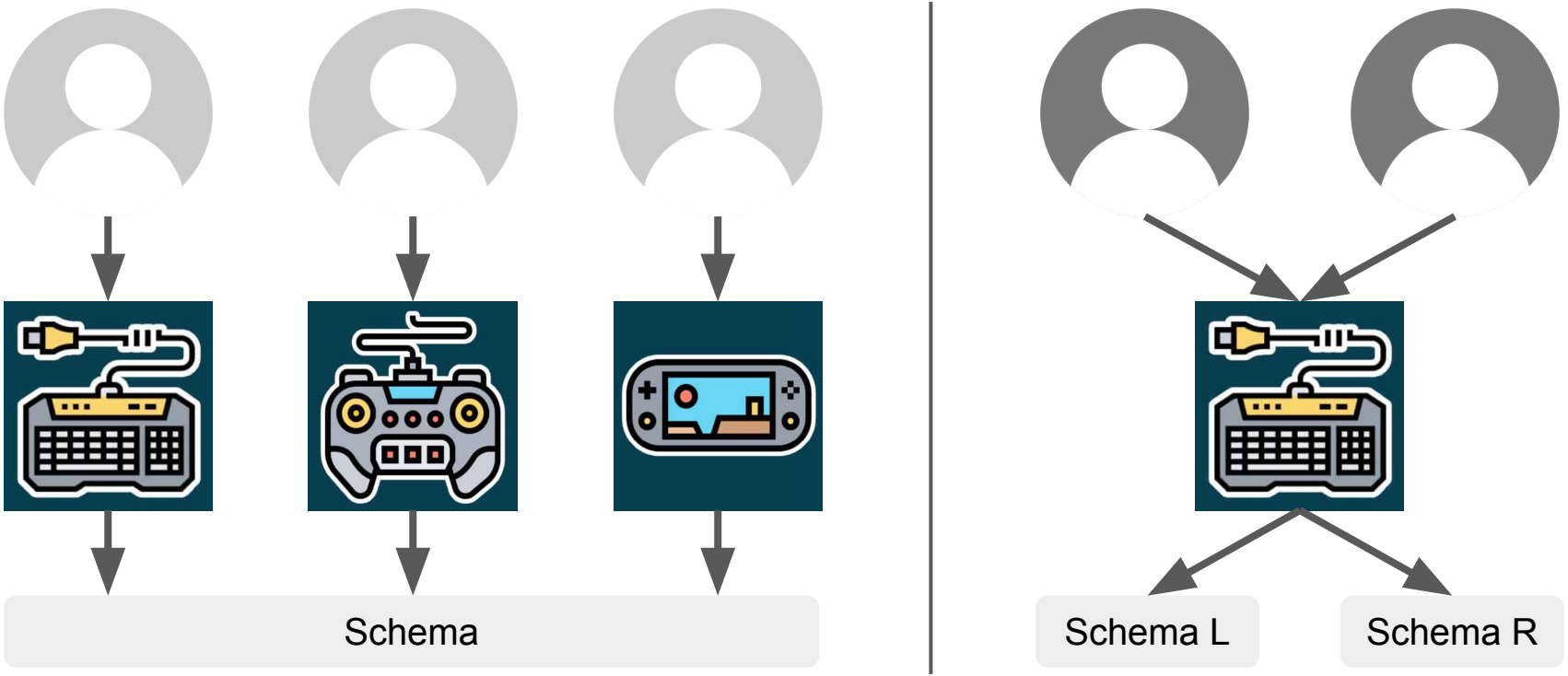
Current Status



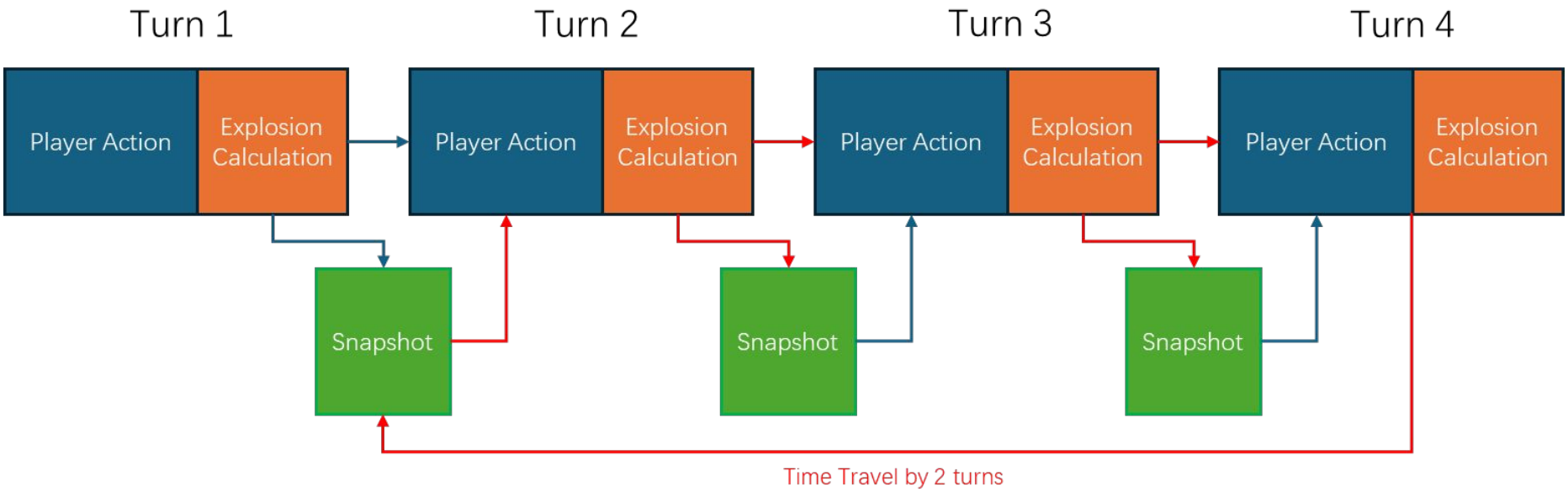
2 Challenges

Keyboard Split | Time Travel | C2.3 | …

2.1 Input System - Keyboard Splitting



2.2 Time Travel Feature



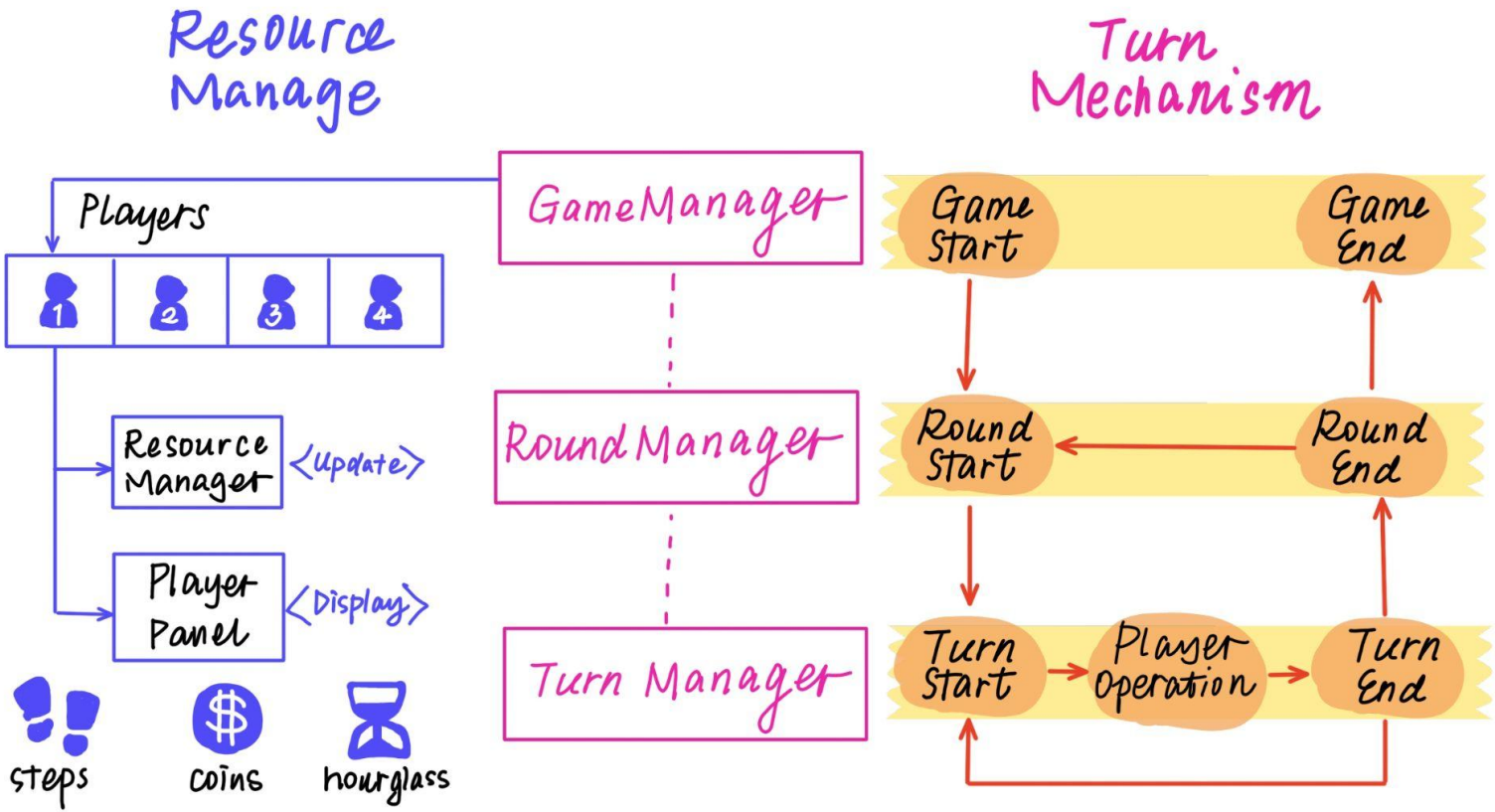
2.2 Time Travel Feature

- Data Structure Definition
  - Use PlayerData, BombData, and ResourceData classes to store the related core data
  - Use Position List to save wall block positions
  - Use GameObject List to allow direct operation on items on map
- Taking Snapshots
  - At the end of every turn, a snapshot of the whole game state will be taken, by storing data to the defined data structure.
- Loading Snapshots
  - Edge Condition: Calling a 3-turn time travel in the 2nd turn.
  - Revert to initialized state when calling an early time travel.
  - Edge Condition: Time Travel back to a turn when the hourglass hasn't been picked up.
  - Make the hourglass an exception in recovery, i.e. the hourglass is a one-time item.

3 Design Revision

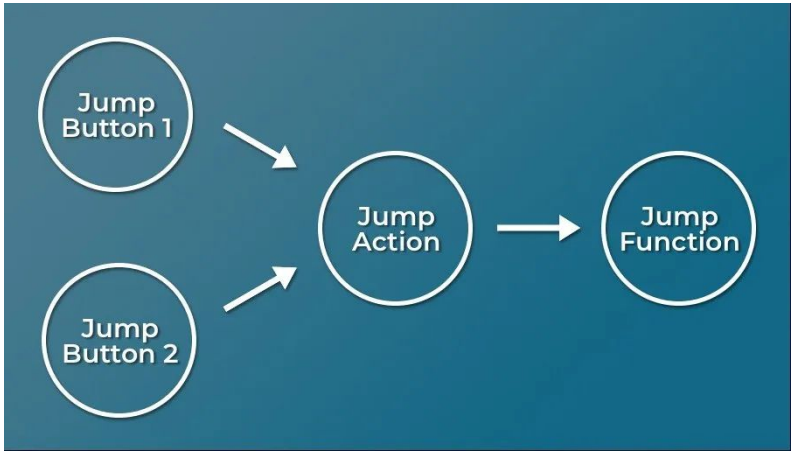
Scalability | D3.2 | D3.3 | …

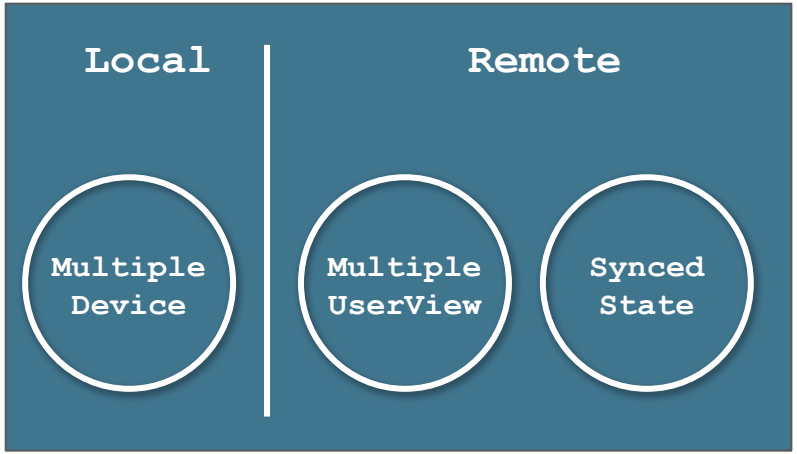
3.1 Scalability - Architecture



3.2 Scalability - User Input Handling

- Support more devices by simple config
- Clear logic with event-driven
- Support local & remote multi-player





3.3 Item Interaction

Current State

- Display of picked-up items (e.g., hourglass, coins, weapons) in a UI panel.
- Visual representations:
  - **Coins**: Yellow sphere.
  - **Hourglass**: Blue sphere.

Item System Overview

- **Item Class**: Defines different item types for in-game usage.
- **Pickup Mechanism**:
  - Uses collision detection.
  - Items are stored in the inventory via a scalable resource management system.

Item Types

- **Stackable Items**: Coins, weapons.
- **Unstackable Items**: Hourglass (unique, one-time use).
    - Ensures fairness: Each player gets a chance to pick up one hourglass per game.

Planned Improvements

1. **Respawn System**:
    - Randomized item respawning.
    - Calibrated with generated maps for each level.
2. **Visual Inventory**:
    - Replace text with icons for items.
    - Enhance player experience with a cleaner UI.
3. **Centralized Explosion Control**:
    - Current explosions are computed on the go
    - Centralize control for having a predictable result

Remaining TODOs

Refactoring Plan

- Current Issue: The Player directly holds references to bombs and walls for interaction.
- Solution: Encapsulate these operations into dedicated functions within the **MapManager** for cleaner, modular code.

Planned Improvements

1. **Improve UI & Assets:**
    - Add better assets and animations for players, bombs, walls, and items.
2. **Multiplayer Optimization:**
    - To reduce delays in remote multiplayer: Compress event/snapshot messages by storing game data (bombs, players, items, maps) in a **K×K×D state matrix**.

3. **Dynamic Map & Item System:**

- Transition from static maps and item placements to:
    - A **respawn system** for items.
    - Procedurally **generated maps** for dynamic gameplay.

Thank you for your time!

Looking forward for your feedback!