Interim Report

Flee Fi Fo From

Team Meeple People

Anastasia Pomelova
Eugene Ghanizadeh Khoub
Mert Ülker
Shyam Rangarajan

Recap

- Digital board game
- Game of chaos and order based on queuing strategy
- Key goals from prototyping:
  - Good networking to substitute for social experience
  - Click and select UI elements instead of drag and drop
  - Split game logic into two parts: perform and check
- **Today: Interim report**

Status Update

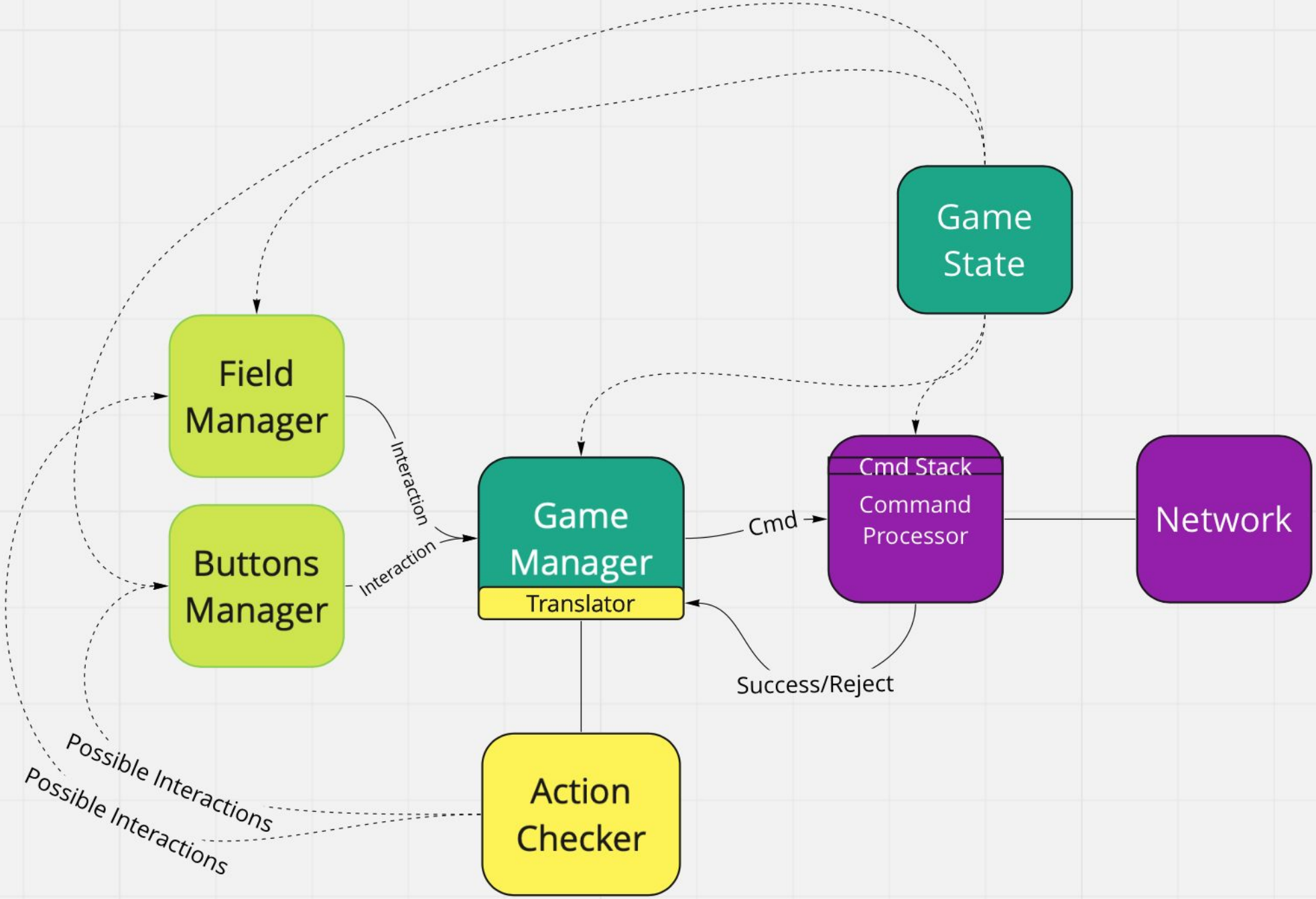
- Design Approach
- Current Status
- Challenges
- Next Steps

Design Approach

- Split into 4 sections
  - User Interaction
  - Game State
  - Game Logic
  - Networking

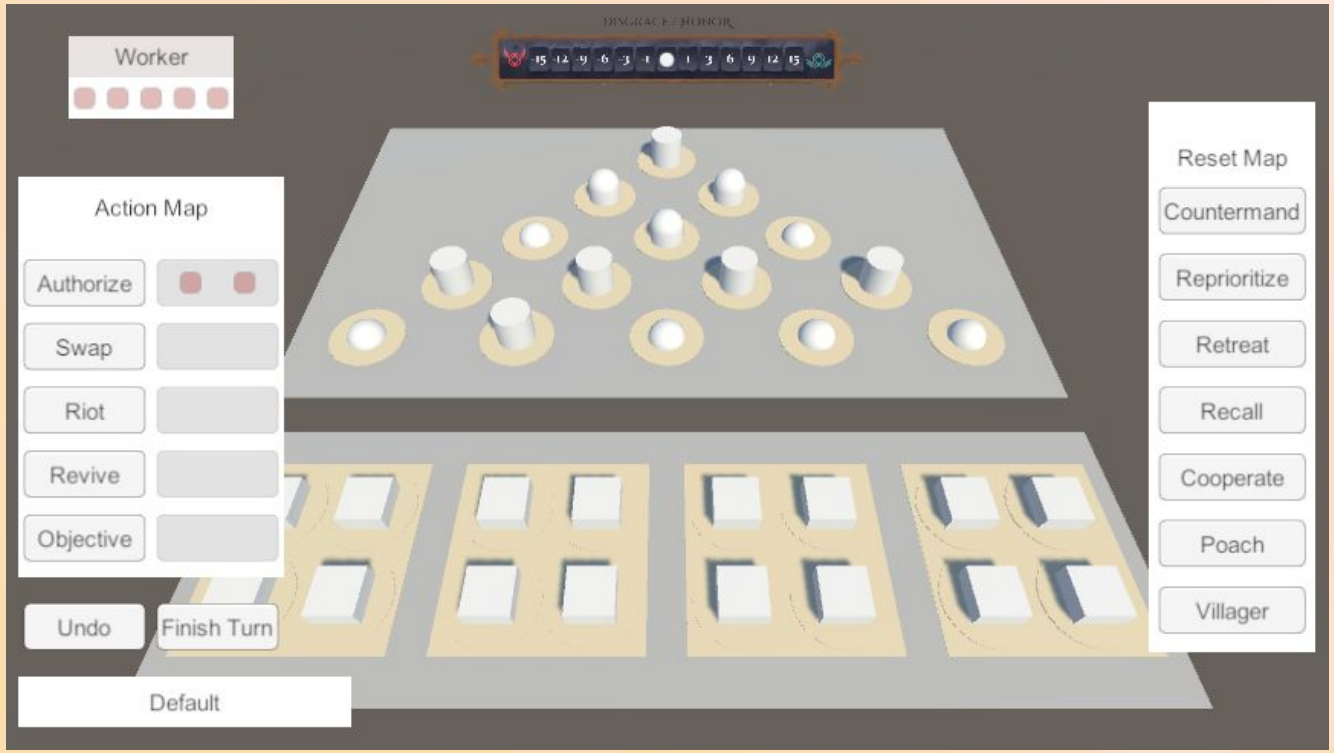
- Why?
  - Game with a lot of rules requires significant prior planning
  - Piecewise development
  - Ability to proceed without dependencies
  - Independent progress without conflicts
  - Merge when stable code base ready

Design Summary



User Interaction

- Interaction with the game board and overlay UI elements
- All visual components needed to play the game (perform part)
- Connection between visuals and logic (check rules part)
- Current State:
  - Base field commands
  - Undo commands
  - Simplified riot



Game State

- Logical representation of the game board (information separation)
- Observable states (i.e. position or health of a piece, which is then Observed by visual layer)
- Helper functions representing some baseline logical connections (neighboring positions, injured pieces on the board, conditional board traversal, etc).

```
private void EnableInjuryBased(bool enableInjured)
{
    GameState.Instance.TraverseBoard(p => {
        var tile = TileByPosition(p);
        if (enableInjured)
        {
            tile.Interactable = GameState.Instance.InjuredVillagerAtPosition(p);
        }
        else
        {
            tile.Interactable = GameState.Instance.HealthyMeepleAtPosition(p);
        }
    });
}
```

```
private void EnableRiotPath(List<Tile> path)
{
    var last = path[path.Count - 1].Position;

    GameState.Instance.TraverseBoard(p => {
        var tile = TileByPosition(p);
        var meeple = GameState.Instance.AtPosition(p);
        tile.Interactable = (
            last.CanMoveTo(p) &&
            (
                meeple == null ||
                (
                    meeple.IsHealthy() &&
                    meeple.GetType() != typeof(DKnight)
                )
            )
        );
    });
}
```

Game Logic

- Adapts the rules of the board game to a code structure
- Rules for each possible action and validity checkers
- Integrates with the game manager
- Current Status:
  - Simplification of challenging rules (and corresponding code req.)
  - Base rules for actions and resets

```
public override void Execute()
{
    base.Execute();
    var meeple = _tile.RemoveMeeple();

    // TODO Authorize: store away piece instead of destroy
    // S.R. Should we use a GameCommand class that contains helper commands
    // TODO: Notes for StoreAway commmand. Will need to redirect piece to correct owner
    //TempStack.add(meeple);
    Destroy(meeple.gameObject);

    if(meeple is Child)
    {
        //TODO: Simplied. If no adult exists in the temp stack
        //if(TempStack.contains(Knight) || TempStack.contains(Commoner) || TempStack.contains(Elder))
        //break;
        //else CurrentPlayer.SelectedWorker.Player.Disgrace();
    }
```

```
public override void CheckFeasibility()
{
    //TODO Step 1: Start loop from tile 1
    /* Psuedo
    if(piece.exists)
    {
        if (piece.injured)
        {
            injuredPiecesOnRow++;
        }
        else
        {
            //TODO: i.e. there exists at least one non-injured piece with line of access
            this.ActionPossible = true;
            break;
        }
```

Game Logic - Rule Changes

- Playtesting feedback:
  - FIFO is visually hard to track
  - Potential for tactile mistakes when tapping
  - Complexity around Authorize action
- Changes:
  - Testing a pivot to an externally tracked FIFO system
  - Based on piece types instead of tapping
  - Authorize specific rules removed, internalized to new FIFO system
- Goal:
  - Simplify the game rules
  - Cleanup exceptions
  - Will correspondingly simplify the implementation of the game logic as well

Networking

- Client-server model realized by employing dedicated server to host the session in the form of headless server build and multiple players connecting the host as clients.
- Server event listener utilized for logging server start and client connection/disconnection
- Connection manager implemented to allow connection over local area network, later to be adjusted for internet connection using a relay server.
- Initial communication manager implemented to publish actions through the server to other clients, utilizing remote procedure calls and executing commands locally on each client.

Challenges

- A game with a lot of rules requires prior planning
  - Our approach: Decide architecture first, decouple into modules instead of layers
  - Commence playtesting of the rule set early on
  - Approach technical achievement of networking early on
- Undo method requirement creates constraints:
  - Our approach: Command pattern
- Challenging ruleset
  - Our approach: Iteratively cleanup rules to ease cognitive (and code) load

Next Steps

- Networking:
  - Adjust system for connection over the internet.
  - Adjust connection manager and corresponding UI to support DNS address and/or server name along with IPv4 address.
  - Implement fallback methods to handle potential server connection errors.
- Game Logic:
  - Verify the modified rules and integrate with code base
- Game Manager:
  - Overall integration
  - Decouple overloaded classes (field, button, action checker)
  - Expand logic for worker rules
- UI:
  - Finish connecting to game logic (pay for actions etc.)
  - Visual improvements

Questions?

Moodle Questions

- Glory Tracker (Honor/Disgrace):
  - Exponential to help set it as a path to victory
  - Makes it harder to snowball via negative actions such as riots, in contrast to linear which is a not a decision point
  - Thematically someone builds up trust over time or is a repeat offender
  - Easier to get disgrace than honor - Similar to real life too where easy to lose face with people?
- Asymmetric abilities?
  - Potentially, but hard to balance. Might consider later as extra
- Objectives same for all?
  - No, hidden objectives per player, so no idea what each person is going for
- Long turns/what to do between turns?
  - Primary aim of the action/reset turn to reduce downtime
  - Involved at least half the time in a 4 player game, or all the time in a 2 player game
- Challenging rules to implement?
  - Potentially, hence staggered approach via target levels