Team Members：

Wang Zihan

Chen Shiyu

Xu Feiyang

Lin Boyi

Huang Zichang

Project Report:

Tactical Chess Strategy Game

Introduction

Our project evolved from the initial proposal of a "Touch-down Game" into a sophisticated chess-like strategy game that innovatively combines terrain mechanics with unique character abilities. While the original proposal focused on a base-capture mechanism, our final implementation created a more dynamic tactical experience through specialized piece abilities and terrain interactions.

Key innovations from the proposal include:

1. Implementation of unique special abilities for each piece type

2. Sophisticated terrain effects that strategically influence piece movement

3. Complex interaction systems between pieces, abilities, and terrains

4. Advanced tactical options through character-specific skills

These changes have transformed the game from a traditional chess variant into a rich tactical experience where player success depends on skillful use of both piece abilities and terrain advantages.

Related Work

Our final implementation draws inspiration from multiple sources:

1. Traditional Chess:

- Basic piece movement patterns
- King capture victory condition
- Turn-based gameplay structure

2. Modern Strategy Games (New Influences):

- Hero-based abilities (like DOTA 2)
- Directional attacks (like Fire Emblem)
- Position-swapping mechanics (like tactical RPGs)
- Area-of-effect abilities (like modern strategy games)

3. Terrain-Based Strategy Games:

- Various terrain effects on movement
- Tactical positioning importance

- Environmental interaction mechanics

Our Work

1. Unique Character Abilities

Each piece type has a distinctive special ability that adds strategic depth:

Queen: Four-directional Strike

* Can attack pieces diagonally in all four directions
* Range of two squares in each direction
* Creates powerful zone control opportunities

Knight: Charge Ability

* Player 1's knights charge downward, Player 2's knights charge upward
* Maximum charge distance is 5 squares
* During charge (once per game):
    - Captures enemy pieces in its path
    - Stops when reaching an ally piece
    - Ignores terrain effects
    - Moves to the last valid position after charge

King: Strategic Swap

* Can swap positions with the nearest friendly knights once per game
* Provides crucial mobility and escape options
* Adds tactical depth to king protection

Bishop: Pawn Summoning

* Can create new pawn pieces at the front position twice per game
* Adds resource generation to tactical options
* Influences board control through unit creation

Bomb: Area Explosion

* Can detonate to affect a 3x3 area
* Eliminates all pieces in the blast radius
* Creates powerful zoning and threat areas
* When capturing or being normally captured, both pieces are destroyed
* Does not trigger mutual destruction when killed by other pieces' abilities

2. Advanced Terrain System

The terrain system creates strategic depth through varied effects:

Mountains:

* Restrict movement to orthogonal single steps
* Create natural choke points
* Influence ability usage and targeting

Forests:

* Allow orthogonal (horizontal/vertical) and diagonal movements of two squares
* Provide strategic positioning opportunities
* Affect line of sight for certain abilities

Rivers:

* Block certain pieces from crossing
* Create natural boundaries
* Influence ability targeting and movement paths

Desert:

* Pieces in the desert cannot capture other pieces
* Affects ability usage and strategic positioning

3. Ability-Terrain Interactions

Special attention was paid to how abilities interact with terrain:

- Knight’s charges is not affected by terrain restrictions
- Bishop's pawn placement must respect terrain rules
- Bomb can use its ability to capture pieces in the desert

Outstanding Features That Deserve High Grades

1. Complex Ability Design:

- Each piece has a unique, tactically significant ability
- Abilities interact meaningfully with terrain
- Balanced power levels between different abilities
- Strategic depth in ability timing and targeting

2. Terrain-Ability Synergy:

- Terrain features affect ability usage
- Strategic positioning becomes crucial
- Multiple layers of tactical decision-making
- Rich interaction between game elements

3. Technical Achievement:

- Robust implementation of complex ability systems
- Smooth integration of terrain effects
- Clean, maintainable code structure

- Comprehensive testing and balancing

4. Version Control Excellence:

- Systematic version management
- Clear commit history and documentation
- Effective branching strategy
- Regular and meaningful commits

5. Team Collaboration:

- Effective task distribution and coordination
- Active communication among team members
- Timely resolution of conflicts and issues
- Productive code review process

Contributions

The project work was distributed among five team members, each focusing on specific core functionalities:

Wang Zihan 123090601(20%)

- Implemented Knight's special charge ability and its movement patterns
- Developed piece piece-finding system

- Created a multiple-piece interaction system for Knight's charge
- Implemented diagonal movement validation patterns
- Added position validation for Knight's special ability
- Developed capture mechanics for Knight's charge ability

Chen Shiyu

- Designed and implemented the Terrain class and terrain types
- Created terrain distribution and map generation system
- Implemented terrain-based movement restrictions
- Added movement validation for terrain interactions
- Developed error handling for illegal moves related to terrain

Xu Feiyang

- Implemented Queen's four-directional attack system
- Created Bomb's explosion mechanics and area effect system
- Developed checkForVictory logic for piece captures
- Implemented piece capture message display system
- Added position validation for Queen and Bomb abilities
- Created the game state tracking for ability usage

Lin Boyi

- Designed and implemented the base Piece class hierarchy
- Created core movement logic and mouse click control system
- Established coordinate system mapping between game board and screen coordinates
- Implemented basic movement validation system in mainwindow.cpp
- Developed the piece selection and movement event handling
- Implemented King's position swap ability

Huang Zichang

- Implemented Bishop's pawn summoning ability
- Created the game board visualization system
- Developed the piece color and appearance system
- Implemented legend display system for pieces and terrain
- Added dynamic board cell coloring based on terrain
- Created ability activation UI and feedback system

Reflections

Major Challenges and Solutions

1.Code Integration：

First, code integration and collaboration proved to be one of our biggest hurdles. Each member was responsible for different components (piece abilities, terrain system, movement framework, etc.), and integrating these components was complex. We addressed this by implementing version control through GitHub for code sharing and management. We also established clear code structure and interface definitions before implementation and held regular code review sessions to ensure compatibility. This experience taught us practical version control and code integration skills that weren't covered in traditional coursework.

2.Establishing a Proper Coordinate System：

Second, we initially struggled with establishing a proper coordinate system for piece movement. This was fundamental to the entire game but proved surprisingly complex. To overcome this, we developed a clear separation between pixel coordinates and game board coordinates, implemented coordinate transformation systems, and created robust validation methods for movement calculations. Through this process, we gained practical experience in spatial system design and coordinate mathematics beyond classroom theory.

3.Deep Integration of Each Member's Work：

Third, unlike previous group projects that focused on presentations where work could be clearly divided, this project required deep integration of each member's work. For example, the member implementing piece abilities needed thorough understanding of

the board system and coordinate framework. We addressed this challenge by establishing weekly two-hour collaboration sessions, implementing regular progress updates and knowledge sharing, and creating detailed documentation for component interfaces. This taught us real-world project management and team communication skills.

4.Code Encapsulation：

Fourth, proper code encapsulation emerged as a crucial factor, particularly in the piece movement system. We learned that well-encapsulated movement functions significantly simplified subsequent development and team handovers. For example, by properly encapsulating the moveTo function in the Piece class, team members working on special abilities could easily implement new movement patterns without worrying about the underlying coordinate transformations or board state management. This made it much easier to add new features and maintain existing code. The encapsulation also made debugging easier as issues could be isolated to specific components. This experience taught us the practical importance of good software engineering principles beyond theoretical classroom knowledge.

Learning Outcomes

Our classroom learning helped us apply fundamental concepts including object-oriented programming principles, GUI development concepts, and basic software architecture principles.

However, the majority of our learning came through self-directed exploration:

1.Version Control and Collaboration:

We learned practical GitHub usage, code merge conflict resolution, and collaborative code review processes.

2.Project Management:

We developed skills in interdependent task management, team coordination in software development, and technical documentation writing.

3.System Design and Encapsulation:

We gained experience in coordinate system design, component interface design, and integration testing strategies.

4.Team Communication:

We improved our abilities in technical knowledge sharing, cross-component coordination, and regular progress synchronization.

Key Takeaways

The project provided valuable experience in real-world software development challenges that aren't typically encountered in classroom settings. The necessity for deep collaboration and integration of different components taught us the importance of clear communication and regular team meetings, understanding dependencies between different system components, proper planning and documentation before implementation, and effective use of version control and collaboration tools.

These experiences have given us practical insights into professional software development practices and team collaboration that will be valuable in our future careers. Most importantly, we learned how to effectively work together on a complex software project where components are deeply interconnected, requiring constant communication and coordination among team members