Cute Territory Game - Project Report

Instructor Name: Ms. Mehak Mazhar

Submitted By:

Muhammad Ali	22K-4691
Muhammad Usman Ghani	22K-4756
Muhammad Faris	22K-4776

Department Of Cybersecurity

1. Introduction

The Cute Territory Game is a grid-based territory capture game built with Pygame. It features a pink aesthetic design theme, multiple game modes, AI opponents, and a power-up system. This report details the technical aspects, design decisions, architecture, and potential improvements for future development.

2. Project Overview

2.1 Game Concept

Players compete to capture territory on a grid by moving over cells, with the goal of having the most territory when the timer expires. The game includes power-ups that provide strategic advantages, such as freezing opponents or gaining bonus points.

2.2 Target Audience

The game targets casual gamers who enjoy:

- Quick play sessions (60-second matches)
- Cute, visually appealing aesthetics
- Simple but strategic gameplay
- Both single-player and AI demonstration modes

3. Technical Implementation

3.1 Architecture Overview

The game is implemented using a state-based architecture with the following main components:

3.1.1 Game States

The game uses an enumeration-based state machine with the following states:

- `MENU`: Main menu for game navigation
- `MODE_SELECT`: Selection of game mode
- `DIFFICULTY_SELECT`: Selection of AI difficulty
- `CUSTOMIZATION`: Player color customization
- `PLAYING`: Active gameplay state
- `GAME_OVER`: End-of-game state showing results

3.1.2 Core Systems

The core systems include:

- **Board Management**: 2D array representing the grid state
- **Player Management**: Tracking positions and scores
- **Power-up System**: Spawning, collecting, and applying effects
- **AI Control System**: Movement decision algorithms with adjustable difficulty
- **Input Handling**: Processing keyboard and mouse events
- **Rendering Pipeline**: Drawing the board, UI elements, and effects
- **State Management**: Controlling game flow between different screens

3.2 Key Technical Features

3.2.1 Grid System

- 20×20 grid implementation
- Cell ownership tracking
- Dynamic territory visualization

3.2.2 AI Implementation

- Tiered decision-making algorithm:
    - Priority for power-up collection
    - Preference for uncaptured/opponent cells
    - Fallback to random valid moves
- Difficulty affects:
    - AI decision quality (smartness parameter)
    - Movement speed
    - Strategic behavior

3.2.3 Power-up System

- Random spawning with position validation
- Timed respawning mechanism
- Effect implementation:
    - Freeze: Time-based movement restriction
    - Points: Immediate score bonus

3.2.4 UI Framework

- Menu system with button highlighting and interaction
- Animated backgrounds for visual appeal
- Information panel with real-time game stats
- Visual feedback for frozen players and power-ups

3.3 Technical Challenges and Solutions

Challenge	Solution
Dynamic power-up spawning	Implemented both timer-based and event-driven spawning with position validation
AI movement speed control	Frame-based movement timer with difficulty adjustment
Player freezing effect	Timestamp-based effect duration with visual indicators
UI state management	Comprehensive state machine with dedicated drawing functions per state
Score balancing	Territory capture with opponent point reduction

4. Design Decisions

4.1 Visual Design

4.1.1 Color Palette

The game uses a consistent pink-based palette to create a cohesive aesthetic:

- Light pink background (`PINK_LIGHT: (255, 200, 230)`)
- Medium pink UI elements (`PINK_MEDIUM: (255, 150, 200)`)
- Dark pink accents (`PINK_DARK: (230, 100, 170)`)
- Hot pink highlights (`HOT_PINK: (255, 50, 150)`)
- Pastel colors for players and elements

4.1.2 UI Layout

- Split-screen design:
    - Main grid area (left)
    - Information panel (right)
- Dedicated menu screens with consistent styling
- Animated backgrounds for visual interest
- Button highlighting for interaction feedback

4.2 Gameplay Design

4.2.1 Game Balance

- 60-second matches for quick play sessions
- Territory scoring with strategic elements:
    - Players lose points when territories are captured
    - Power-ups provide temporary advantages
- AI difficulties provide appropriate challenge levels:
    - Normal: 70% optimal decision-making
    - Hard: 90% optimal decision-making with faster movement

4.2.2 Player Progression

- Score-based success metric
- Real-time feedback through territory visualization
- End-game results screen with winner declaration

5. Testing and Quality Assurance

- Manual testing of all game states and transitions
- AI behavior testing across difficulty levels
- Edge case testing for power-up spawning and collection
- Performance testing for animation smoothness

6. Future Development Opportunities

6.1 Short-term Improvements

- Add sound effects for actions (movement, capture, power-up collection)
- Implement additional power-up types
- Add option to customize game duration
- Improve AI pathfinding for more strategic play

6.2 Long-term Features

- Network multiplayer support
- Level editor for custom maps
- Campaign mode with progressive challenges
- Achievement system
- Persistent high scores
- Additional visual themes

6.3 Technical Enhancements

- Optimize rendering for larger grid sizes
- Implement sprite-based graphics instead of shape primitives
- Refactor into more modular class-based architecture
- Add unit testing framework

7. Conclusion

The Cute Territory Game successfully implements a casual, visually appealing territory capture game with strategic elements. The state-based architecture provides a solid foundation for future enhancements, while the current feature set delivers an engaging gameplay experience with good replayability through different modes and difficulty settings.

The combination of simple mechanics with strategic depth makes the game accessible while still providing challenge. The visual design creates a cohesive and appealing aesthetic that enhances the casual, fun nature of the gameplay.

8. Code Structure

```
game.py
  ├── Constants and Initialization
  ├── Game State Enumerations
  ├── Core Game Variables
  ├── Game Mechanics Functions
  │    ├── spawn_power_up()
  │    ├── move_agent()
  │    ├── check_power_ups()
  │    └── reset_game()
  ├── Rendering Functions
  │    ├── draw_board()
  │    ├── draw_scores_and_timer()
  │    ├── draw_menu()
  │    ├── draw_mode_select()
  │    ├── draw_difficulty_select()
  │    ├── draw_customization()
  │    └── game_over_screen()
  └── Main Game Loop
       └── game_loop()
```

9. References

- **Pygame Documentation**: https://www.pygame.org/docs/
- **Python Standard Library**: https://docs.python.org/3/library/
- **Game Programming Patterns** by Robert Nystrom: https://gameprogrammingpatterns.com/