CEN Deliverable 1: Tiger Zone

I. System Artifacts

The system in question of design is a game called Tiger Zone which will be modeled after Carcassonne.

1. **UML**

UML diagrams are designed for the UI, game play, and AI. See attached diagrams for a comprehensive understanding.

1. **Outline**

In addition, an outline of the objects, rules, and scoring for game play are described below. This outline will help the team during the implementation process.

*Objects*

Tiles

Global Properties

4 edges, 4 corners used to determine Terrain location

Part of stack from which players will choose a Tile at random

Can have multiple Terrains on one Tile

Terrain

Cloister

Whole Tile - can have a Road

Continuity

Independent

Complete when

Surrounded by Tiles

Follower

Monk

Score

Completed: 9 points

Incomplete: 9 - <number of open adjacent spots>

City

Segmented

Continuity

Continuous along edges

Wall

Border separating City and other Terrains

Tiles that are entirely Cities have no walls

Extends from corner to corner

Can have multiple Walls per Tile

Tile edge containing City and no Wall is continuous from that edge

Pennant

Flag that allows City Tiles to be worth more points

Complete when

City interior continuous, City Wall forms loop

Follower

Knight

Score

Completed: 2 points per Tiles containing City, 2 points per pennant within City

Incomplete: 1 points per Tiles containing City, 1 points per pennant within City

Road

Segmented

Continuity

Continuous along edges

Path through Tile

Starts on one edge and ends on another

Separates Farms

Complete when

Continuous Road capped by Crossing/City/Cloister or Road forms loop

Follower

Thief

Score

Completed: 1 point per Tiles containing Road, including caps

Incomplete: 1 point per Tile containing continuous Road

Crossing

Whole Tile

Meeting of 3-4 Roads

Caps Roads

(Extends Road?)

Field/Farm

Segmented

Continuity

Continuous along edges

Default Terrain - Any Terrain that is not a City/Road/Cloister is a Farm

Complete when

Game ends

Follower

Farmer

Score

Completed: 3 points per completed City that borders Farm

Incomplete: N/A

Followers

7 per player

Placed on Terrains

One Follower per continuous Terrain

Terrains can combine and cause multiple followers to belong to a Terrain

Follower Types

Knight

Placed on Cities

Returned when City is complete and scored

Thief

Placed on Roads

Returned when Road is complete and scored

Farmer

Placed on Farms

Does not Return

Monk

Placed on Cloisters

Returned when Cloister is complete and scored

*Basic Rules*

Players place a Tile each turn

Players can place Followers on tiles to earn points

Designated starting placed first

Each player has 7 Followers to play

Turn order

1. Draw and place Tile

Random Tile from stack

Tile must be adjacent to another Tile edge

Tile terrain must be continuous with adjacent Tile terrain

Tiles with no legal placement are discarded, a new Tile is drawn

2. Deploy a Follower (optional)

1 Follower can be deployed per turn

Can only deploy from unused Followers in supply

Can only deploy Follower on Tile placed this turn

Follower type is specified by Tile terrain placed on

Only one follower per continuous Tile Terrain Segment

If player has no Followers, this part is skipped

3. Score completed Cloisters, Roads, Cities

Completing a Road

Conditions: Road capped on 2 ends by Crossing/City/Cloister, or if Road forms complete loop

Player with Thief on completed road gains points for each Tile that makes it up

Completing a City

Conditions: City is continuous and surrounded by City Wall

Player with Knight in City collects 2 points per Tile that makes up the City

If a Pennant is in a City Segment, the player earns 2 more points per pennant when the City is complete

Completing a Cloister

Conditions: Cloister is completely surrounded by Tiles (8)

Player with Monk in Cloister earns 9 points (Cloister Tile + surrounding Tiles)

After Terrain Segment is scored, Followers are returned to respective players to be used again

Placing a Follower on a Terrain Segment completed that turn allows immediate scoring

1. Complete Terrain Segment

2. Deploy appropriate Follower

3. Score the Terrain Segment

4. Return Follower

If a Tile is added to a Segment and the result is multiple Followers in a completed Segment, the points go to the player with more Followers in that segment. Ties result in points for both players.

*Endgame Scoring*

Points go to player with most Followers on a Segment. Ties function the same as with completed segments.

Incomplete Roads/Cities

1 point per Tile of a continuous Segment.

Pennants are worth 1 extra point in cities.

Incomplete Cloisters

1 point per adjacent tile

Farms

Not scored until the end of the game

Farmers remain in place all game

Farms separated by Roads, other Terrain types

Farms must border completed Cities in order to earn points

Player with most Farmers in a field earns 3 points for each complete City bordering the Farm. Ties handled normally

Player with the most points wins

II. Development Planning Artifacts

1. **Overall Project Plan**

The overall project plan is organized as dictated by the chart. The iterations are broken down into sections as follows: UI, gameplay, and AI. Each iteration includes implementation and testing of each user requirement. The testing criteria is detailed in the Testing Artifacts. The team will meet briefly before the start of the iteration phase to discuss and delegate tasks. Then team will confer over tasks and issues throughout the week. The Systems Artifacts describe the objects, rules, and scoring each part of iteration implementation must be aware of and actively involve in the coding process.

|  |  |  |
| --- | --- | --- |
| **Phase** | **Start** | **Finish** |
| Design and planning | 10/31 | 11/6 |
| Iteration 1- UI | 11/7 | 11/11 |
| Iteration 2 - Game Play | 11/12 | 11/19 |
| Iteration 3 - AI | 11/20 | 11/27 |
| Demonstration | 11/28 | 11/28 |

**B. Iteration Plan**

In each iteration, the user requirements and system requirements are detailed. The user requirements state the system services and operational constraints for the sole purpose of the customer- the customer being the user or player of the game. The system requirements are written for client and contractor- the client being the company, or in this case Professor Smalls and the contractor being UF- to detail descriptions of system services and operational constraints.

|  |  |  |
| --- | --- | --- |
| **Iteration** | **User Requirements** | **System Requirements** |
| 1: UI | * Tiles * Game board * Meoples * Player avatars | * GUI game board appears similar to Carcassonne board * GUI tiles appear similar to Carcassonne tiles * GUI Meoples appear similar to Carcassonne user pieces * GUI displays tile for user to see. * Tiles able to be placed on board. * Board recognizes valid tile placement. * Board recognizes valid Meople placement. * GUI recognizes user clicks for tile placement. * GUI recognizes user clicks for Meople placement. * Player avatars must be lit and change according to that player’s turn. |
| 2: Game Play | * Follow game play and scoring of Carcassonne | * See Systems Artifacts for rules and scoring * Runs two games for demonstration purposes |
| 3: AI | * Understand game play * Understand game scoring * Maximizes game point play | * Makes valid tile placement * Makes valid Meople placement * Maximizes point play - offensive strategy * Minimizes point gains of opponent - defensive strategy * Balances short term point gains with long term point gains * Balances defensive strategies with offensive ones |

III. Testing Artifacts

1. **Testing Strategy**

Objectives of testing will be to ensure user interface, AI, and gameplay follow the desired outcome of the client. The team will take on a behavior driven testing strategy. The plan is detailed below.

1. **Testing Plan**

This table outlines per iteration what behavior the team will test for and the actions taken to ensure that behavior is followed.

|  |  |  |
| --- | --- | --- |
| **Iteration** | **Behavior to be tested** | **Actions to test** |
| 1: UI | * Valid placement of tiles * Valid placement of Meoples * Player Avatar highlights and changes according to turn * Tiles are displayed to user * GUI recognizes all valid user clicks * GUI appropriates handles incorrect user clicks | * Simulate valid and invalid clicking for tile placement * Simulate valid and invalid clicking for Meople placement * View every tile to be shown to user during play * Watch that player avatar highlight is changed upon turn |
| 2: Game Play - Rules | * Follows rules of Carcassonne | * User simulates invalid rule-following * Tries to break each rule given to document outcome |
| 2: Game Play - Scoring | * Follows scoring of Carcassonne | * User documents throughout game point gains in comparison to computer generated point gains |
| 2: Game Play - Run Time | * Runs two games at once | * Upon run two GUI are generated * User can play similar games in both GUIs |
| 3: AI | * Maximizes points gained - offensive strategy * Minimizes pointed gained by the opponent - defensive strategy * Balances short term point gains with long term point gains * Balances defensive strategies with offensive ones | * Runs through simulation using only maximize point algorithm * Runs through simulation using only minimize opponents points gained algorithm * Runs using both algorithms * Runs through simulation with short term point gain algorithm * Runs through simulation with long term point gain algorithm * Runs using both algorithms * Compare AI results to user’s choice during that instance of game play for the whole game |

1. **Test Case**

The test case in question is the game Carcassonne. If it looks and plays like Carcassonne without error or invalid gameplay then the game will have passed the test. The test case for the AI will be the AI as demonstrated in class by Professor Smalls. If it plays like it played in the demonstration, the AI will have passed the test.

1. **Test Report**

Any bugs found after a test will be documented and taken to the team to evaluate and correct. Once corrected, the team will run back through the testing plan for that specific iteration.