OBJECT-ORIENTED PROGRAMMING HY-252

PROJECT 2015

Stefanou - Stamatiadis Stefanos: ΑΜ2843

Σχεδιασμός Stratego Game

Η υλοποίηση βασίζεται στο μοντέλο MVC (Model View Controller). Ο Controller ειναι υπεύθυνος για τη λειτουργεία του παιχνιδιού. Λαμβάνει δεδομένα (πχ κινησεων των παικτών) απο τον Viewer και μετά απο επεξεργασία τους (πχ είναι νόμιμη μία κίνηση ή το παιχνίδι εφτασε στο τέλος του) τροποποιεί κατάλληλα τα Models.

To Controller package (stratego.controller)

περιέχει την κλάση StrategoAppController που περιγράφει τον controller του παιχνιδιού καθώς και την κλάση StrategoAppRunner που περιέχει το entry point της εφαρμογής (main method). O StrategoAppController περιέχει τις εξής μεθόδους:

* public void start()
  + Initializes the StrategoAppController: 1. Creates a Board 2. Creates two players 3. Creates the Viewer for the game.
* public void insertToken2Board(Board b, Token t, int row, int col)
  + Insert token reference to board.
* public Token getReference2Token(Board b, int row, int col)
  + Precondition: Initialized Board and Players.
* public List<Vector2D> movementSystemScout(Board board, Scout sct)
  + Implements Scout moving/attack model.
* public List<Vector2D> powerMovementSystemFlight(Board board, Dragon drg)
  + Implements Dragon moving/attack model.
* public List<Vector2D> powerMovementSystemSpeed(Board board, MovablePlayerToken mtkn)
  + Implements Speed special movement/attack model.
* public List<Vector2D> powerMovementSystemRange(Board board, MovablePlayerToken mtkn)
  + Implements Range special attack model.
* public List<Vector2D> tokenSelection(Board board, Token tkn)
  + Precondition: The Token to exist in Board. Computes valid movement/attack locations for the token in board.
* public List<Vector2D> tokenSpecialSelection(Board board, Token tkn)
  + Precondition: Token to be on board and belong to special Tokens. Computes valid movement/attack locations for the token in board.
* public boolean isInsideBoard(Vector2D pos)
  + Return true if position falls inside the board bounds. Else returns false.
* public boolean hitsRock(Vector2D pos)
  + Precondition: Position must be inside board bounds. Return true if position falls on top of rocks, on board. Else returns false.
* public List<Vector2D> validInBoard(List<Vector2D> loc)
  + Check if Board position is valid by game terms.
* public List<Vector2D> validInPlayer(Board board, List<Vector2D> loc)
  + Filter valid movements in order to not conflict with same player Tokens
* public List<Vector2D> validForAttack(Board board, List<Vector2D> loc)
  + Valid classic (cross-like) attack positions for each position in input list, that enemy Token exists.
* public List<Vector2D> positionsAttackOptions(Board board, Vector2D pos)
  + Checks if input position has valid attack options.
* public boolean tokenAction(Board board, Token src, Token trg, Vector2D trgloc, StrategoAppViewer viewer, boolean specialPower)
  + Performs an action from source token to the target token. Actions can be: 1. Nothing (action not permitted) 2. Move (valid movement action) 3. Attack (valid attack action)
* public boolean saveToken(Board board, MovablePlayerToken tkn)
  + Saves the lost (input) Token and places it back to the board in the first empty position. If no empty positon Token is not saved.
* public void convertToken(Board board, PlayerToken tkn)
  + Converts the input PlayerToken to the other side. The particular token can not be converted back, till the end of the game
* public void takeToken(Board board, PlayerToken tkn)
* public void moveToken(Board board, MovablePlayerToken src, Vector2D trgloc)
* public void attackToken(Board board, MovablePlayerToken src, PlayerToken trg, boolean specialPower)
* public boolean isThereNextMovePossible(Board board, Turn turn)
  + Precondition:Must be called after a tokenAction have been made. Check if next move is impossible.
* public boolean isThereDropedFlag(Turn turn)
  + Precondition:Must be called after a tokenAction have been made. Check if dropped flag.
* public Player doWeHaveAWinner(Board board, Turn turn)
  + Check if winning condition.
* public void endGame(Turn turn)
  + Ends the game and prints the winner's' name
* public Turn getTurn()
  + Returns the Turn Object of the game

To Model package (stratego.model)

περιέχει τις παρακάτω κλάσεις/interfaces:

* Token (interface)
* PlayerToken (abstract class)
* BackgroundToken (abstract class)
* ImmovablePlayerToken (abstract class)
* MovablePlayerToken (abstract class)
* Player (abstract class)
* Fire
* Ice
* Grass
* Rock
* Κλάσεις για όλα τα πιόνια των παικτών

To View package (stratego.view)

περιέχει την κλάση StrategoAppViewer με τις μεθόδους:

* public StrategoAppViewer(Board StrategoBoard)
  + Κάνει preload τα εικονίδια των πιονιών σε ένα hashmap και δημιουργεί ένα παράθυρο με τόσα buttons όσα τα πιόνια στον πίνακα παιξίματος.
* private JButton getGridButton(int r, int c); Accessor
  + Επιστρέφει την αναφορά στο button της θέσης row/col.
* private JButton createGridButton(final int row, final int col)
  + Δημιουργεί ένα Button κάνοντας override το function public void actionPerformed(ActionEvent e) ώστε να επιτρέπει την αλληλεπίδραση με το χρήστη (highlight selected token, call controller to perform token action etc).
* private JPanel createGridPanel()
  + Δημιουργεί το view του πίνακα παιξίματος.
* private void updateGrid()
  + κάνει update τον πίνακα παιξίματος μετά απο input του χρήστη.
* private void toggleButton(JButton b, boolean toggle)
  + Toggles selected/non selected state στα πιονια (πχ γιατι υπαρχουν πιόνια που δεν επιτρεπουν την αλληλεπιδραση πχ πιόνια αλλου παικτη/ rock πιόνια)
* private ImageIcon getTokenIcon(Token tkn)
  + Επιστρέφει το εικονίδιο του δοθέν πιονιού απο το hashmap.
* private ImageIcon getIconImage(String filename)
  + Επιστρεφει την εικόνα του εικονιδίου με αυτό το όνομα αρχείου.
* private void selectToken(int row, int col)
  + Ενεργοποιεί το πιόνι μετά απο input του χρήστη μέ ότι αυτό συνεπάγεται ανάλογα με το current state του πίνακα παιξίματος.
* public void display()
  + Εμφανίζει τον πίνακα παιξίματος