

Entities & Responsibilities

GameWorld: Manages game initialisation & admin tasks (e.g game config, game setup, main game loop, player turn.)

Tile: Represents the tiles which the dragons will interact with (stand on)

TileDrawData: Data class for organising data required for drawing any tile (associated with Tile abstract class).

PlayableEntity: Represents the playable entity a player interacts with

GameBoard: Represents the game board. It runs the interactions with the game board by the players (e.g performing movement, flipping chit card)

ChitCard: Represents the chit cards and their effects

EventBus: Handles registration of listeners, and notification of appropriate listeners on event fire

WinEventXxxx.....: Publisher and listeners for win event

MoveActionXxxx.....: Publisher and listeners for a move action (for characters) that is fired

DrawableByAsset: Indicates that the object is drawable by pygame using assets

DrawAssetInstruction: A data class for organising data required for drawing an asset

ModularClickableSprite: Allows classes to be represented as a sprite that is clickable on a screen.

Patterns Used

Observer: WinEventPublisher, WinEventListener

- Why?: Don't have to check all starting tiles to see if win occurred. Allows for wins from other sources

Singleton: EventBus

- Why?: Should be one central event bus managing all events

Todo

Cardinalities

Starting tiles must be winning tiles for DefaultGameBoard (need to make the typing more strong, extra class inheriting from Tile [WinnableTile])

Notes

Upcasts are safe
Circular dependencies = too many responsibilities
<https://softwareengineering.stackexchange.com/questions/306483/how-to-solve-circular-dependency>
Java supports circular dependencies
https://www.reddit.com/r/ProgrammingLanguages/comments/yvkysht/languages_which_support_circular_dependency/

DefaultGameBoard.__init__: First parameter is the sequence of tiles as their identifiers. Second parameter is [(starting tile, tile # to connect to in first parameter, playable character to start in it)]. The second parameter (starting tiles) will be used to make insertions to main tile sequence property.
DefaultGameBoard.starting_tiles: [starting tiles]
DefaultGameBoard.move_character_by_steps: Moves a dragon from its tile by x number of steps.
DefaultGameBoard.flip_chit_card: Flip a chit card on this game board. Chit card object will be obtained by click listener in pygame.
DefaultGameBoard.add_chit_card: Add a chit card to the game board

ChitCard.__init__: Create a chit card with the # number of symbols
ChitCard.perform_effect: Perform the chit card's effect on the dragon.
Run by flipChitCard
ChitCard.set_flipped: Set flipped

ModularClickableSprite.__init__: Run update click hitboxes
ModularClickableSprite.rect: Dictionary mapping click hit boxes to the object associated with it
ModularClickableSprite.on_click: Do something to the object on click
ModularClickableSprite.update_click_hitboxes: Update click hitboxes based on draw instructions (get_draw_clickable_assets_instructions)

DrawableByAsset.get_draw_assets_instructions: Gets the drawing instructions for drawing an object onto the pygame screen using assets

GameWorld.__init__: Configure and initialise playable characters, game board & its tiles & its chit cards.
GameWorld.run: Contains the main game loop. Handles policy of player turns, and drawing logic (based on drawable)

MoveActionPublisher.notifySubscribers: Notify subscribers about a move action that needs to be handled. Character and steps to take passed in

Tile.__init__: Initialise tile with playable character (dragon) and animal
Tile.place_character: Place dragon onto tile and perform related functionalities
Tile.remove_character: Remove the dragon that's on the tile (if any)
Tile.character_on_tile: Returns the dragon on the tile (if any)
Tile.get_draw_asset_instructions: Return empty array if tile draw data is none (i.e don't draw)

Dragon.take_turn: Allows a dragon to take its turn & perform actions (e.g chit card flipping) by interacting with screen

CaveTile.place_character: Place dragon onto tile and perform check whether it was return back to own cave, trigger win

EventBus.shared: Get the eventbus singleton
EventBus.publishWinEvent: Publish win event onto bus
EventBus.addListener: Add a listener for a certain event type

WinEventListener.onPlayerWin: On a player win, do something

WinEventPublisher.notifySubscribers: Notify subscribers about the player who won