|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| main | None | None | This is the main function which will run the game. |

### AStar Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| calculate\_f\_value | coords: List | int | This function will calculate the f-value of the RectNode at the given coordinates. The f-value will be calculated by adding the weight of the RectNode at the coordinates given (we can get this information by passing in the coordinates given into the get\_weight\_at\_node method in self.rect\_array\_obj) with the heuristic we have calculated for the RectNode at the coordinate (we can get the heuristic for the coordinate by passing the coordinate as a key into the heuristic\_dict dictionary). We will then return this f-value. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the AStar class. |
| run | None | Stack | Runs the A\* pathfinding algorithm and saves the coordinates of the checked nodes in the checked\_nodes stack and the coordinates of the path nodes in the path stack. |

### AnimationManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| add\_coords\_to\_animation\_dict | coords: List, animation\_type: AnimationTypes, foreground\_color: pygame.Color, background\_color: pygame.Color, speed: float | None | Adds coordinates of nodes which need to be animated to the self.animation\_dict dictionary. |
| add\_ui\_element\_to\_ui\_element\_interpolation\_dict | ui\_element\_type: ColorUITypes, initial\_colour: pygame.Color, final\_colour: pygame.Color | None | Allows for a specific UI Color to be animated to a different colour by adding it to the self.ui\_element\_interpolation\_dict dictionary. |
| update\_border\_and\_board\_interpolation | None | Dictionary | If the border or board colour is being animated then this function will return the colour which needs to be drawn in the current frame. |
| update\_ui\_element\_interpolation\_dict | None | List | This will return a list containing the colours which should be drawn on the current frame of each UI Color which is being animated. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray | None | Initalizes the AnimationManager class. |
| update\_coords\_animations | background\_color: pygame.Color | None | Will update and draw each node which is being animated in the self.animation\_dict dictionary. |
| interpolate\_board\_or\_border | interpolation\_type: AnimationTypes, initial\_color: pygame.Color, final\_color: pygame.Color, speed: float | None | Adds the colours which need to be animated for either the board or the border to self.board\_or\_border\_interpolation\_dict dictionary. |

### AnimationNode Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_init\_\_ | coords: List, animation\_type: AnimationTypes, foreground\_color: pygame.Color, background\_color: pygame.Color, speed: float, column\_width: float, row\_width: float, row\_width\_int: int, grid\_height\_offset: int | None | Initializes the AnimationNode class. |

### BFS Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the BFS (Breadth First Search) class. |
| run | None | Stack | Runs the BFS (Breadth First Search) pathfinding algorithm and saves the coordinates of the checked nodes in the checked\_nodes stack and the coordinates of the path nodes in the path stack. |

### BidirectionalBFS Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| find\_common\_coord | None | List | This function will return the first coordinate it finds which is in both the search\_a\_checked\_nodes stack and the search\_b\_checked\_nodes stack. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the BidirectionalBFS class. |
| run | None | Stack | Runs the Bidirectional BFS (Best First Search) pathfinding algorithm and saves the coordinates of the checked nodes in the checked\_nodes stack and the coordinates of the path nodes in the path stack. |

### Client Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| update\_pathfinding\_algorithm\_speed | pathfinding\_algorithm\_speed: int | List | If the changed\_pathfinding\_algorithm\_speed attribute is set to True, this function will set the changed\_pathfinding\_algorithm\_speed attribute to False and return a list containing 2 elements which is the boolean value True and the self.pathfinding\_algorithm\_speed attribute. However, if the changed\_pathfinding\_algorithm\_speed attribute is set to False, we will return a list containing the boolean value False and pathfinding\_algorithm\_speed variable we have been given. |
| create\_network\_event | event\_type: NetworkingEventTypes, args: Tuple | None | If the connected\_to\_server attribute is set to True, this function will send the data for the event we are creating to the server. This will be done by first identifying what NetworkingEventType the event\_type variable we have been given is. Once we know which NetworkingEventType we are going to send we will create a dictionary called command which will contain a key which will be the event\_type and the value for the key will the tuple args we are given if it is not empty, if args is empty we will simply make the value None. We will then turn the dictionary into json and send the json data to the server. However, if the event\_type was NetworkingEventTypes.DISCONNECT\_FROM\_SERVER then we will also want to shut down the socket after sending the command to the server and set the connected\_to\_sever attribute to False. |
| handle\_server\_events | None | None | This function will be run in a separate thread. It will run a loop which will continuously check for new messages from the server as long as the connected\_to\_server attribute is set to True. If it receives and event from the server it will turn the json string into a dictionary and interpret the NetworkingEventType the dictionary contains. Depending upon what the NetworkingEventType is the function will then apply the event. |
| reset\_cancel\_recursive\_division | None | None | This function will set the cancel\_recursive\_division and recursive\_division\_cut\_off\_point attributes to False. |
| apply\_resolution\_divider | None | None | This function will run the set\_resolution\_divider method in self.screen\_manager and pass in the self.resolution\_divider attribute so that resolution\_divider attribute in self.screen\_manager has been updated. We will then recreate the grid using the reset\_rect\_array method in self.rect\_array\_obj. If the current\_pathfinding\_algorithm attribute is not equal to None, we will also reset the path and checked nodes stacks using the reset\_path\_pointer and reset\_checked\_nodes\_pointer methods respectively. Additionally, if the current\_maze\_generation\_algorithm is not equal to None we will also reset the maze\_pointer attribute using the reset\_maze\_pointer method. |
| update\_recursive\_division\_speed | recursive\_division\_speed: int | List | If the changed\_recursive\_division\_speed attribute is set to True, this function will set the changed\_recursive\_division\_speed attribute to False and return a list containing 2 elements which is the boolean value True and the self.recursive\_division\_speed attribute. However, if the changed\_recursive\_division\_speed attribute is set to False, we will return a list containing the boolean value False and recursive\_division\_speed variable we have been given. |
| connect\_to\_server | server\_ip\_address: Str, port: int | bool or None | If the connected\_to\_server attribute is set to False, this function will try to establish TCP connection with the server at the IP Address and Port Number given. If we are able to successfully connect to the server we will set the connected\_to\_server attribute to True and start a new Thread running the handle\_server\_events method. However, if we are unable to connect to the server we will return False. |
| update\_current\_pathfinding\_algorithm | pathfinding\_algorithm: An instance of a child class of the PathfindingAlgorithm class. | List | If the changed\_current\_pathfinding\_algorithm attribute is set to True, this function will set the changed\_current\_pathfinding\_algorithm attribute to False and return a list containing 2 elements which is the boolean value True and the self.current\_pathfinding\_algorithm attribute. However, if the changed\_current\_pathfinding\_algorithm attribute is set to False, we will return a list containing the boolean value False and current\_pathfinding\_algorithm variable we have been given. |
| \_\_init\_\_ | screen\_manager: ScreenManager, grid: Grid, rect\_array\_obj: RectArray, pathfinding\_algorithms\_dict: Dict, maze\_generation\_algorithms\_dict: Dict, animation\_manager: AnimationManager, events\_dict: Dict, color\_manager: ColorManager | None | Initalizes the Client class. |
| reset\_cancel\_pathfinding\_algorithm | None | None | This function will set the cancel\_pathfinding\_algorithm to False. |
| update\_current\_maze\_generation\_algorithm | maze\_generation\_algorithm: An instance of a child class of the MazeGenerationAlgorithm class. | List | If the changed\_current\_maze\_generation\_algorithm attribute is set to True, this function will set the changed\_current\_maze\_generation\_algorithm attribute to False and return a list containing 2 elements which is the boolean value True and the self.current\_maze\_generation\_algorithm attribute. However, if the changed\_current\_maze\_generation\_algorithm attribute is set to False, we will return a list containing the boolean value False and current\_maze\_generation\_algorithm variable we have been given. |
| update\_resolution\_divider | resolution\_divider: int | List | If the changed\_resolution\_divider attribute is set to True, this function will set the changed\_resolution\_divider attribute to False and return a list contain 2 elements which is the boolean value True and the self.resolution\_divider attribute. However, if the changed\_resolution\_divider attribute is set to False, we will return a list containing the boolean value False and resolution\_divider variable we have been given. |

### ColorManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| extract\_rgb\_color\_from\_pygame\_color | color: pygame.Color | Tuple | Will take in a pygame.Color instance and extract the rgb values of the pygame.Color instance into a tuple and it will then return the tuple. |
| set\_dark\_theme | None | None | Will set the current theme to be the default 'Dark Theme'. |
| delete\_custom\_theme\_from\_themes\_list | name: Str | None | This function will delete a custom theme from self.themes\_list dictionary if the custom theme exists. It will also save any changes done the self.themes\_list dictionary to the data/themes/themes.json file using the save\_themes\_list method. |
| set\_theme\_colors\_dict | new\_theme\_colors\_dict: Dict | None | Will replace the contents of the theme colors dictionary with the dictionary provided. |
| get\_all\_custom\_theme\_names\_from\_themes\_list | None | List | This function will go through the self.themes\_list dictionary search for any custom themes and append the names of the custom themes to a list and then return that list. |
| save\_themes\_list | None | None | This function will json.dumps method to save all of the information in the self.themes\_list into the data/themes.themes.json file. |
| get\_theme\_from\_themes\_list | name: Str | Dict or None | This function will return the theme dictionary of the theme with the specified name from the self.themes\_list dictionary if it exists (otherwise it will return None). |
| check\_custom\_theme\_exists | name: Str | Bool | This function will search the self.themes\_list dictionary to see if a custom theme with the specified name exists if it does the function will return True or else the function will return False. |
| save\_theme\_to\_themes\_list | name: Str, colors\_dict: Dict | None | This function will add a new theme to the self.themes\_list dictionary, and it will save all of these changes to the data/themes/themes.json file using the save\_themes\_list method. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, animation\_manager: AnimationManager | None | Initialises the ColorManager class. |
| set\_light\_theme | None | None | Will set the current theme to be the default 'Light Theme'. |
| get\_theme\_colors\_dict | None | Dict | Will return the theme colors dictionary. |
| set\_and\_animate\_dark\_theme | current\_pathfinding\_algorithm: An instance of a child class of the PathfindingAlgorithm class. | None | Will set the current theme to be the default 'Dark Theme' and also animate the theme using the set\_and\_animate\_theme\_colors\_dict method. |
| do\_custom\_themes\_exists | None | Bool | This function will search the self.themes\_list dictionary for any custom themes, if a custom theme exists it will return True or else it will return False. |
| set\_node\_color | node\_type: Int, color: pygame.Color | None | Will set the specified node type (either from ColorNodeTypes or ColorUITypes) with the specified color in the theme colors dictionary. |
| set\_and\_animate\_light\_theme | current\_pathfinding\_algorithm: An instance of a child class of the PathfindingAlgorithm class. | None | Will set the current theme to be the default 'Light Theme' and also animate the theme using the set\_and\_animate\_theme\_colors\_dict method. |
| load\_themes\_into\_themes\_list | None | None | This function will read all the json data containing information about themes in the data/themes/themes.json file and then load it into the self.themes\_list dictionary. |
| set\_and\_animate\_node\_color | node\_type: Int, color: pygame.Color, current\_pathfinding\_algorithm: An instance of a child class of the PathfindingAlgorithm class. | None | This function will set the specified node type in the theme colors dictionary with the specified colour if it is different from the colour the node type already has. Additionally, if the colour is different then it will be animated based upon what node type it has been set to: If the node type is either ColorNodeTypes.BOARD\_COLOR or ColorNodeTypes.BORDER\_COLOR then the interpolate\_board\_or\_border method from the instance of the Animation Manager will be used. If it is any other value in ColorNodeTypes then the add\_coords\_to\_animation\_dict method from the instance of the AnimationManager will be used. If the node type is in ColorUITypes then the add\_ui\_element\_to\_ui\_element\_interpolation\_dict method from the instance of the AnimationManager will be used. |
| set\_and\_animate\_theme\_colors\_dict | new\_theme\_colors\_dict: Dict, current\_pathfinding\_algorithm: An instance of a child class of the PathfindingAlgorithm class. | None | Will replace the theme colours dictionary with the dictionary provided and any colours which are different from the original theme colours dictionary will be animated (this will be done by using the set\_and\_animate\_node\_color method from the instance of the AnimationManager class). |
| get\_theme\_color | type: Int | pygame.Color | This function will return a colour of specified type from the theme colours dictionary as a pygame.Color object. |
| get\_all\_theme\_names\_from\_themes\_list | None | List | This will go through all the themes in the self.themes\_list dictionary and append their names to a list and then return that list. |

### DFS Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the DFS class. |
| run | None | Stack | Runs the DFS (Depth First Search) pathfinding algorithm and saves the coordinates of the checked nodes in the checked\_nodes stack and the coordinates of the path nodes in the path stack. |

### Dijkastra Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the Dijkastra class. |
| run | None | Stack | Runs the Dijkastra pathfinding algorithm and saves the coordinates of the checked nodes in the checked\_nodes stack and the coordinates of the path nodes in the path stack. |

### FontManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| delete\_custom\_ui\_font | font\_name: Str | None | If a font with the font\_name given exists as a custom font in the font\_json\_data dictionary (we can get a list of all the custom fonts in the font\_json\_data dictionary using the get\_custom\_ui\_font\_names method), we will first check if the font\_name given is the same as the current\_font\_name attribute. If it is we will set the current\_font\_name attribute equal to the default font we load at the start of the game (by default I use the 'Roboto' font) and we will also run the set\_current\_font method and pass in the name of the default font we use as a string. After this, we will remove the key-value pair associated with the font\_name given, from the font\_json\_data dictionary, and then we will run the save\_font\_json\_data method. |
| get\_ui\_font\_names | None | List | This function will return a list containing all the keys in the font\_json\_data dictionary. |
| custom\_ui\_fonts\_exist | None | bool | This function will go through each font in the font\_json\_data dictionary. It will then retrieve the dictionary which is the value of the font and check if the custom\_font key in this dictionary contains the value True. If it does we will return True and exit the function otherwise we will continue to the next iteration. If we have completed all the iterations then this would mean that there aren't any custom fonts in the font\_json\_data dictionary, and so we will return False. |
| get\_custom\_ui\_font\_names | None | List | This function will go through each font in the font\_json\_data dictionary. It will then retrieve the dictionary which is the value of the font and check if the custom\_font key in this dictionary contains the value True. If it does we will add this font to the custom\_font\_names list. After we have iterated over all the fonts in the font\_json\_data dictionary we will return the custom\_font\_names list. |
| set\_normal\_font\_size | font\_size: int | None | This function will set the ui\_normal\_font\_size\_value attribute to be equal to the font\_size given, we will then perform the relevant operations to ensure that the UI also uses this font size for every element on the screen except for the titles at the top of windows. |
| \_\_init\_\_ | manager: pygame\_gui.ui\_manager.UIManager, theme\_manager: pygame\_gui.core.interfaces.appearance\_theme\_interface.IUIAppearanceThemeInterface, theme\_json\_data: Dict | None | Initializes the FontManager class. |
| add\_ui\_font | font\_name: Str, normal\_font\_regular\_weight\_file\_path: Str, normal\_font\_bold\_weight\_file\_path: Str, title\_font\_regular\_weight\_file\_path: Str | None | If the font with the font\_name specified does not exist as a key in the font\_json\_data dictionary (we can get a list of all the keys in the font\_json\_data dictionary using the get\_ui\_font\_names method) we will add a new key-value pair into the dictionary where the key will be the font\_name given and the value will be another dictionary containing the following key-value pairs: { "custom\_font": True, "regular": normal\_font\_regular\_weight\_file\_path, "bold": normal\_font\_bold\_weight\_file\_path, "title": title\_font\_regular\_weight\_file\_path } After this we will run the save\_font\_json\_data method, and then we will load the font files into the UI, so we can the font whenever we want. |
| set\_current\_font | font\_name: Str | None or False | If a font with the font\_name specified exists as a key in the font\_json\_data dictionary (we can get a list of all the keys in the font\_json\_data dictionary using the get\_ui\_font\_names method) we will the set the current\_font\_name attribute to be equal to the font\_name given, and we will perform the relevant operations to ensure that the UI is using this new font. In the case that the font with the font\_name specified does not exist in the font\_json\_data dictionary we will return False. |
| save\_font\_json\_data | None | None | Will turn the font\_json\_data dictionary into json and save it in the data/fonts/font\_info.json file. |
| load\_ui\_fonts | None | None | This function will go through each font in the font\_json\_data dictionary and check if the file paths of the font files still exist in the filesystem. If any of the font files no longer exist in the filesystem we will add the font's name to the fonts\_to\_remove list. If all the font files for the font exist we will load the font files into the UI, so we can use the font whenever we want. After we finish going through all the fonts in the font\_json\_data dictionary we will iterate over all the fonts in the fonts\_to\_remove list and remove the key-value pair in the dictionary which is associated with the font. After we have finished iterating through the fonts\_to\_remove list we will run the save\_font\_json\_data method. |

### GameUIManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| update\_pathfinding\_algorithm\_speed | pathfinding\_algorithm\_speed: int | None | This function will set the value of self.pathfinding\_algorithm\_speed to the pathfinding\_algorithm\_speed given and also set the value of the pathfinding algorithm speed slider to be the same as the pathfinding\_algorithm\_speed given. |
| get\_current\_maze\_generation\_algorithm | None | An instance of a child class of the MazeGenerationAlgorithm class. | This function will return the maze geneartion algorithm object which is stored as the value of the key self.current\_maze\_generation\_algorithm in the maze\_generation\_algorithms\_dict dictionary. |
| handle\_ui\_window\_open | None | bool | This function will return True if there are any windows open in the game otherwise it will return False. |
| build\_ui\_running\_pathfinding\_algorithm\_state | None | None | This method will set the state attribute to GameUIStates.UI\_RUNNING\_PATHFINDING\_ALGORITHM\_STATE and perform the required operations to ensure that the UI is in GameUIStates.UI\_NORMAL\_STATE. |
| generate\_networking\_menu | kill\_networking\_menu: bool | None | This function will first check if the kill\_networking\_menu variable given is set to True if it is we will destroy the current networking menu. After this we will create a new networking menu. |
| build\_ui\_running\_recursive\_division\_state | None | None | This method will set the state attribute to GameUIStates.UI\_RUNNING\_RECURSIVE\_DIVISION\_STATE and perform the required operations to ensure that the UI is in GameUIStates.UI\_RUNNING\_PATHFINDING\_ALGORITHM\_STATE. |
| not\_normal\_state | None | bool | This function will check if the state attribute is equal to the GameUIStates.UI\_NORMAL\_STATE, if it is we will return False otherwise we will return True. |
| handle\_ui\_text\_entry\_finished\_event | event: pygame.event | None | This function will handle the pygame\_gui.UI\_TEXT\_ENTRY\_FINISHED event for the GameUIManager, ThemeWindow and the SettingsWindow. |
| get\_weight | None | int | Getter for the weight attribute. |
| update\_current\_maze\_generation\_algorithm | maze\_generation\_algorithm: An instance of a child class of the MazeGenerationAlgorithm class., is\_server\_event: bool | None | This function will set the current\_maze\_generation\_algorithm attribute to the maze\_generation\_algorithm given. After this we will destroy the maze generation algorithm menu and create a new one. We will also run the build\_ui\_running\_recursive\_division\_state method if the is\_server\_event variable given is set to True and the maze\_generation\_algorithm given is set to be MazeGenerationAlgorithmTypes.RECURSIVE\_DIVISION. |
| get\_recursive\_division\_speed | None | int | Getter for the recursive\_division\_speed attribute. |
| update\_resolution\_divider | None | None | This function will set the value of the resolution divider slider to be the same as the resolution\_divider attribute in self.screen\_manager. |
| set\_ui\_colours\_from\_current\_theme | None | None | This function will go through each ui node type in ColorUITypes and call the set\_colour\_to\_ui\_element method on it, we will set the colour argument to be the colour in the theme for that ui node type (we can get this colour by using the get\_theme\_color method in self.color\_manager and pass in the ui node type we are currently iterating over as the argument) and we will also set the update\_theme argument to False. |
| get\_current\_pathfinding\_algorithm | None | An instance of a child class of the PathfindingAlgorithm class. | This function will return the pathfinding algorithm object which is stored as the value of the key self.current\_pathfinding\_algorithm in the pathfinding\_algorithms\_dict dictionary. |
| handle\_ui\_window\_closed\_event | event: pygame.event | None | This function will handle the pygame\_gui.UI\_WINDOW\_CLOSE event for the GameUIManager, ThemeWindow, UINetworkingManager, SettingsWindow and the TutorialWindow. |
| update\_current\_pathfinding\_algorithm | pathfinding\_algorithm: An instance of a child class of the PathfindingAlgorithm class., heuristic: PathfindingHeuristics, is\_server\_event: bool | None | This function will set the current\_pathfinding\_algorithm attribute to be equal to the pathfinding\_algorithm given. This function will then destroy the current pathfinding algorithms menu and the heuristics menu and create a new pathfinding algorithms menu and heuristics menu depending upon the pathfinding\_algorithm and heuristic given. After this we will check if the is\_server\_event variable given is set to True. If it is we will run the build\_ui\_running\_pathfinding\_algorithm\_state method. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager, grid: Grid, client: Client, server: Server, pathfinding\_algorithms\_dict: Dict, maze\_generation\_algorithms\_dict: Dict, events\_dict: Dict | None | Initializes the GameUIManager class. |
| get\_cursor\_node\_type | None | CursorNodeTypes | Getter for the cursor\_node\_type attribute. |
| get\_pathfinding\_algorithm\_speed | None | int | Getter for the pathfinding\_algorithm\_speed attribute. |
| cancel\_pathfinding\_algorithm | None | None | This function will reset the path pointer and checked nodes pointer for the current pathfinding algorithm, as well as setting the timer for the DRAW\_CHECKED\_NODES and DRAW\_PATH events to 0. The function will then run the build\_ui\_normal\_state method. |
| generate\_theme\_menu | theme: Str or None, kill\_theme\_menu: Bool | None | If the theme variable given is set to None we will set the theme variable to be equal the current\_theme\_name attribute in self.color\_manager. After this we will check if the kill\_theme\_menu variable given is set to True if it is we will destroy the current theme menu. After this we will create a new theme menu. |
| run\_pathfinding\_algorithm | pathfinding\_algorithm: An instance of a child class of the PathfindingAlgorithm class., heuristic: PathfindingHeuristics | None | This function will be used to run a pathfinding algorithm, and it will do the following things in the order given: 1) Run the reset\_rect\_array\_adjacent\_nodes method in self.rect\_array\_obj. 2) Run the gen\_rect\_array\_with\_adjacent\_nodes method in self.rect\_array\_obj. 3) Run the reset\_non\_user\_weights method in self.rect\_array\_obj. 4) Run the reset\_path\_pointer method in pathfinding\_algorithm. 5) Run the reset\_checked\_nodes\_pointer method in pathfinding\_algorithm. 6) Run the reset\_animated\_checked\_coords\_stack method in pathfinding\_algorithm. 7) Run the reset\_animated\_path\_coords\_stack method in pathfinding\_algorithm. 8) Set the heuristic attribute in pathfinding\_algorithm to be the same as the heuristic given. 9) Run the run method in pathfinding\_algorithm. |
| update\_recursive\_division\_speed | recursive\_division\_speed: int | None | This function will set the value of self.recursive\_division\_speed to the recursive\_division\_speed given and also set the value of the recursive division speed slider to be the same as the recursive\_division\_speed given. |
| cancel\_recursive\_division | cut\_off\_point: int or None | None | This function will run the cut\_maze method on the current maze generation algorithm and pass in the cut\_off\_point variable given. It will then set the timer for the DRAW\_MAZE event to 0 and then call the build\_ui\_normal\_state method. |
| create\_heuristics\_menu\_with\_distances | starting\_value: Str | None | This function will first destroy the current heuristics menu. After this it will check if the starting\_value given is equal to the string 'Manhattan Distance' or 'Euclidean Distance' and it will set the heuristic attribute to be PathfindingHeuristics.MANHATTAN\_DISTANCE or PathfindingHeuristics.EUCLIDEAN\_DISTANCE accordingly. After this we will create a new heuristics menu with the new value of the heuristics attribute. |
| draw | None | None | This function will draw of the elements in the UI onto the screen. |
| run\_current\_maze\_generation\_algorithm | None | None | This function will first run the reset\_path\_pointer and reset\_checked\_nodes\_pointer methods on the current pathfinding algorithm (we can access the current pathfinding algorithm by accessing the object stored as the value of the key which is self.current\_pathfinding\_algorithm in the pathfinding\_algorithms\_dict dictionary). We will also run the reset\_maze\_pointer method on the current maze generation algorithm (we can also access the current maze generation algorithm by accessing the object stored as the value of the key which is self.current\_maze\_generation\_algorithm in the maze\_generation\_algorithms\_dict dictionary). After this we will run the maze generation algorithm specified by self.current\_maze\_generation\_algorithm. |
| handle\_bottom\_ui\_drop\_down\_menus\_open | None | bool | If the marked or weighted node menu, clear nodes menu or theme menu are open we will return True otherwise we will return False. |
| handle\_ui\_colour\_animations | None | None | This function will first call the update\_ui\_element\_interpolation\_dict method in self.animation\_manager and then iterate over each key-value pair in this dictionary (this key value pair will contain the ui node type and the colour of the ui node type). We will then call the set\_colour\_to\_ui\_element method and pass in the ui node type and colour. After this we will call the set\_node\_color method in self.color\_manager and pass in the ui node type and the colour. |
| handle\_ui\_border\_width\_changed | None | None | If the changed\_ui\_border\_width attribute in self.settings\_window is set to True, we will set it to False and then perform the required operations to ensure that the all the UI Elements are using the correct border width. |
| build\_ui\_normal\_state | None | None | This method will set the state attribute to GameUIStates.UI\_NORMAL\_STATE and perform the required operations to ensure that the UI is in GameUIStates.UI\_NORMAL\_STATE. |
| set\_colour\_to\_ui\_element | ui\_node\_type: ColorUITypes, colour: Tuple, update\_theme: bool | None | This function will set the colour of the ui\_node\_type specified to the colour specified. If the update\_theme variable given is set to True we will also update these changes so that they can be seen in the UI. |
| handle\_ui\_selection\_list\_dropped\_selection | event: pygame.event | None | This function will handle the pygame\_gui.UI\_SELECTION\_LIST\_DROPPED\_SELECTION for the GameUIManager, ThemeWindow and the SettingsWindow. |
| create\_empty\_heuristics\_menu | None | None | This function will first destroy the current heuristics menu, and it will set the heuristic attribute to None. After this it will create a new heuristics menu. |
| handle\_ui\_color\_picker\_color\_picked\_event | event: pygame.event | None | This function will handle the pygame\_gui.UI\_COLOUR\_PICKER\_COLOUR\_PICKED for the GameUIManager and the ThemeWindow. |
| handle\_ui\_button\_pressed\_event | event: pygame.event | None | This function will handle the pygame\_gui.UI\_BUTTON\_PRESSED\_EVENT event for the GameUIManager, ThemeWindow, SettingsWindow and the TutorialWindow. |
| handle\_ui\_horizontal\_slider\_moved\_event | event: pygame.event | None | This function will handle the pygame\_gui.UI\_HORIZONTAL\_SLIDER\_MOVED event for the GameUIManager and the SettingsWindow. |
| handle\_ui\_file\_dialog\_path\_picked\_event | event: pygame.event | None | This function will handle the pygame\_gui.UI\_FILE\_DIALOG\_PATH\_PICKED for the GameUIManager and the SettingsWindow. |
| handle\_ui\_selection\_list\_new\_selection | event: pygame.event | None | This function will handle the pygame\_gui.UI\_SELECTION\_LIST\_NEW\_SELECTION for the GameUIManager, ThemeWindow, and the SettingsWindow. |
| update\_client\_received\_new\_theme | None | None | This function will first check if the received\_new\_theme attribute in self.client is equal to True, if it is we will set it to False then we will call the generate\_theme\_menu method and set the kill\_theme\_menu argument to True. We will then create a message window telling the user that another client in the server has sent them a new theme. |
| update\_networking\_server\_connection\_broken | None | None | This function will first check if the server\_connection\_broken attribute in self.client is equal to True, if it is we will set it to False then we will call the server\_connection\_has\_broken method in self.ui\_networking\_manager. We will then create the networking menu again using the generate\_networking\_menu method, and we will also create the theme menu again using the generate\_theme\_menu method with the kill\_theme\_menu argument being set to True. |
| handle\_ui\_drop\_down\_menu\_changed\_event | event: pygame.event | None | This function handles the pygame\_gui.UI\_DROP\_DOWN\_MENU\_CHANGED event for the GameUIManager, ThemeWindow and SettingsWindow. |

### GreedyBFS Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the GreedyBFS class. |
| run | None | Stack | Runs the Greedy BFS (Best First Search) pathfinding algorithm and saves the coordinates of the checked nodes in the checked\_nodes stack and the coordinates of the path nodes in the path stack. |

### Grid Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| reset\_marked\_nodes | animate: bool | None | This function will go through each RectNode in self.rect\_array\_obj.array and check if the 'marked' attribute is set to True. If the attribute is set to True then it will be set to False and the node will be animated using the add\_coords\_to\_animation\_dict in self.animation\_manager if the animate variable given is set to True. |
| mark\_end\_node | node: RectNode | None | This function will find the current RectNode which has the 'is\_end\_node' attribute set to True in self.rect\_array\_obj.array and it will set it to False. It will then set the 'is\_end\_node' attribute of the RectNode passed into the function to True, and then animate both the old and new end nodes using the add\_coords\_to\_animation\_dict in self.animation\_manager. |
| draw\_grid | None | None | This function will draw the lines for the grid onto the screen. The number of columns and rows on the grid should be the same as the values of the num\_of\_columns and num\_of\_rows attributes in self.screen\_manager. The width between each column and row should also be the same as the values of the column\_width and row\_width attributes in self.screen\_manager. |
| get\_board\_info | None | List | This function will go through each RectNode in self.rect\_array\_obj.array and get information about the start node's coordinates, the end node's coordinates, a list containing the coordinates of all the nodes with the 'marked' attribute set to True and a list containing the coordinates of all the nodes with the 'is\_user\_weight' attribute set to True. All this information will be added to a list along with the value of the resolution\_divider attribute in self.screen\_manager. The function will then return this list. |
| unmark\_node\_at\_mouse\_pos | mouse\_coords: Tuple | None | This function will find the RectNode in self.rect\_array\_obj.array whose rect collides with the mouse coordinates given, it will then pass this RectNode into the unmark\_node method. |
| unmark\_node | node: RectNode | None | If the RectNode given has the 'is\_user\_weight' attribute set to True then the 'is\_user\_weight' attribute will be set to False and the 'weight' attribute will be set to 1, the node will then be animated using the add\_coords\_to\_animation\_dict method in self.animation\_manager. However, if the RectNode given has the 'marked' attribute set to True then the 'marked' attribute will be set to False and the node will also be animated using the add\_coords\_to\_animation\_dict method in self.animation\_manager. |
| reset\_all\_weights | animate: Bool | None | This function will go through each RectNode in self.rect\_array\_obj.array and check if the 'is\_user\_weight' attribute is set to True. If the 'is\_user\_weight' attribute is set to True then it will be set to False and the 'weight' attribute will be set to 1. The node will then be animated using the add\_coords\_to\_animation\_dict in self.animation\_manager if the animate variable given is set to True. |
| mark\_end\_node\_at\_mouse\_pos | mouse\_coords: Tuple | None | This function will find the RectNode in self.rect\_array\_obj.array whose rect collides with the mouse coordinates given, it will then pass this RectNode into the mark\_end\_node method. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the Grid class. |
| mark\_start\_node\_at\_mouse\_pos | mouse\_coords: Tuple | None | This function will find the RectNode in self.rect\_array\_obj.array whose rect collides with the mouse coordinates given, it will then pass this RectNode into the mark\_start\_node method. |
| mark\_node | node\_type: CursorNodeTypes, node: RectNode, weight: int | None | Depending upon the node\_type given the RectNode given will be marked as either a marked node or a weighted node. If node\_type is CursorNodeTypes.MARKED\_NODE then the RectNode will have it's 'marked' attribute set to True and it will be animated using the add\_coords\_to\_animation\_dict method in self.animation\_manager. If node\_type is CursorNodeTypes.WEIGHTED\_NODE then the RectNode will have it's 'is\_user\_weight' attribute set to True and the 'weight' attribute will be set the to weight we have been given. This node will also be animated using the add\_coords\_to\_animation\_dict method in self.animation\_manager. |
| mark\_start\_node | node: RectNode | None | This function will find the current RectNode which has the 'is\_start\_node' attribute set to True in self.rect\_array\_obj.array and it will set it to False. It will then set the 'is\_start\_node' attribute of the RectNode passed into the function to True, and then animate both the old and new start nodes using the add\_coords\_to\_animation\_dict in self.animation\_manager. |
| mark\_node\_at\_mouse\_pos | mouse\_coords: Tuple, node\_type: CursorNodeTypes, weight: int | None | This function will find the RectNode in self.rect\_array\_obj.array whose rect collides with the mouse coordinates given, it will then pass this RectNode along with the node\_type and weight given into the mark\_node method. |
| draw\_rect\_nodes | None | None | This function will draw the different types of nodes onto the screen. It will do this by going through each RectNode in self.rect\_array\_obj.array and depending upon which attributes are set to True in the RectNode they will be drawn with their relevant colours onto the screen (you will be able to get these colours using self.color\_manager). The priority of these attributes are as follows: 1) is\_start\_node 2) is\_end\_node 3) marked 4) is\_user\_weight I have highlighted this order of properties because a RectNode may have more than one of these attributes set to True at the same time (for example a RectNode might have the is\_start\_node attribute and the marked attribute both set to True, in this case the node should be drawn as the start node because that is the attribute with the higher priority). If this is the case then the node will be drawn as the attribute it has set to True with the highest priority. |

### MazeGenerationAlgorithm Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| reset\_maze\_pointer | None | None | This function will set the value of the self.maze\_pointer attribute to be -1. |
| reset\_animated\_coords\_stack | None | None | This function will set the value of the self.animated\_coords attribute to be a new stack which will have a size that is the same as the number of nodes in the grid (we can calculate this by multiplying the num\_of\_rows attribute by the num\_of\_columns attribute which are both found in self.screen\_manager. |
| reset\_maze | None | None | This function will set the value of the self.maze attribute to be a new stack which will have a size that is the same as the number of nodes in the grid (we can calculate this by multiplying the num\_of\_rows attribute by the num\_of\_columns attribute which are both found in self.screen\_manager. |
| draw | None | None | This function is called every frame when we need to draw a maze generation algorithm. When we first start drawing a maze generation algorithm the self.maze\_pointer attribute is set to 0. Over time, we increment the value of self.maze\_pointer (this process is handled separately and not by this function) and start drawing and animating nodes onto the screen. We will first check if the coordinate in self.maze at the index of self.maze\_pointer is in the self.animated\_coords stack. If it is not we will animate the coordinate using the add\_coords\_to\_animation\_dict method in self.animation\_manager, and we will also push these coordinates onto the self.animated\_coords stack. After this we will then draw all the other coordinates which came before the coordinate at self.maze\_pointer as rectangles (since they will have been animated before). |
| cut\_maze | cut\_off\_index: int or None | None | This function is used to change the number of coordinates in the self.maze stack. If the cut\_off\_index variable has a value of None we will then make the cut\_off\_index variable have the same value as the self.maze\_pointer attribute. If the cut\_off\_index variable has been given a value other than None then we will set the self.maze\_pointer attribute to have the same value as the cut\_off\_index variable. We will then resize the self.maze stack by removing any coordinates which come after the index of the cut\_off\_index variable. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initalizes the MazeGenerationAlgorithm class. |
| update\_maze\_pointer | None | int | This function will increment the value of self.maze\_pointer as long as it is less than the size of the self.maze stack (we can get the size of self.maze using self.maze.get\_size()). If this condition is met we will increment self.maze\_pointer and return 0 otherwise we will return -1. |

### PathfindingAlgorithm Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| get\_path | None | Stack or None | If the path attribute is an empty stack this function will return None, otherwise it will return the path attribute. |
| get\_manhattan\_distance | coords: List, end\_node\_coords: List | int | This function will calculate the Manhattan distance between two coordinates. |
| reset\_animated\_path\_coords\_stack | None | None | This function will set the value of the self.animated\_path\_coords attribute to be a new stack which will have a size that is the same as the number of nodes in the grid (we can calculate this by multiplying the num\_of\_rows attribute by the num\_of\_columns attribute which are both found in self.screen\_manager. |
| update\_checked\_nodes\_pointer | None | int | This function will increment the checked\_nodes\_pointer attribute and return 0 if the checked\_nodes\_pointer attribute is smaller than the total size of the checked\_nodes stack (we can get the total size of the stack by using the get\_size method in self.checked\_nodes), otherwise we will return -1. |
| get\_euclidean\_distance | coords: List, end\_node\_coords: List | int | This function will calculate the Euclidean distance between 2 coordinates. |
| reset\_animated\_checked\_coords\_stack | None | None | This function will set the value of the self.animated\_checked\_coords attribute to be a new stack which will have a size that is the same as the number of nodes in the grid (we can calculate this by multiplying the num\_of\_rows attribute by the num\_of\_columns attribute which are both found in self.screen\_manager. |
| reset\_path\_pointer | use\_checked\_nodes\_foreground\_color: Bool | None | This function will first check if the reset\_path\_nodes attribute is set to False and the path\_pointer attribute is not equal to -1. If this is true we will loop through all the coordinates in the path stack up to the index of the path\_pointer attribute. In each iteration we will first check if the use\_checked\_nodes\_foreground\_color variable is set to True, if it is we will then check if the coordinate is in the checked\_nodes stack, and if this condition is also true we will then animate the coordinates using the add\_coords\_to\_animation\_dict method in self.animation\_manager with the background colour being ColorNodeTypes.CHECKED\_NODE\_FOREGROUND\_COLOR, and we will move onto the next iteration from there. If the iteration has not been continued we will animate the coordinates with the normal background colour. After this, we will set the path\_pointer attribute to -1, the reset\_path\_nodes attribute to True, and we will also reset the path stack. |
| update\_path\_pointer | None | int | This function will increment the path\_pointer attribute and return 0 if the path\_pointer attribute is smaller than the total size of the path stack (we can get the total size of the stack by using the get\_size method in self.path), otherwise we will return -1. |
| draw | None | None | This function is called every frame when we need to draw a pathfinding algorithm. When we first start drawing a pathfinding algorithm the self.checked\_nodes\_pointer attribute is set to 0. Over time, we increment the value of self.checked\_nodes\_pointer (this process is handled separately and not by this function) and start drawing and animating checked nodes onto the screen. Once the checked\_nodes\_pointer is equal to the total size of the checked\_nodes stack (we can get the size of the stack using the get\_size method in self.checked\_nodes) we will then set the drawn\_checked\_nodes variable to True, and then we will start to draw and animate path nodes in the same way we drew and animated the checked nodes, but instead we will use the path\_pointer attribute and path stack. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the PathfindingAlgorithm class. |
| get\_checked\_nodes | None | Stack or None | If the checked\_nodes attribute is an empty stack this function will return None, otherwise it will return the checked\_nodes attribute. |
| reset\_checked\_nodes\_pointer | check\_for\_colliding\_path\_nodes: Bool | None | This function will first check if the reset\_checked\_nodes attribute is set to False. If it is then we will loop through all the coordinates in the checked\_nodes stack up to the index of the checked\_nodes\_pointer attribute. In each iteration we will first check if the check\_for\_colliding\_path\_nodes variable is set to True, if it is we will check if the coordinate is in the path stack and if it is not in the path stack we will animate it using the add\_coords\_to\_animation\_dict in self.animation\_manager. If the check\_for\_colliding\_path\_nodes variable is set to False we will animate the coordinates directly instead of checking if it is in the path stack. After this we will set the checked\_nodes\_pointer attribute to -1, the reset\_checked\_nodes attribute to True, and we will also reset the checked\_nodes stack. |

### PriorityQueue Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_str\_\_ | None | Str | Whenever we try to print out the instance of the PriorityQueue class it will instead print the self.heap list. |
| \_\_iter\_\_ | None | Iterator | Makes it so that when we loop over the instance of the PriorityQueue class it will actually loop over the self.heap list. |
| replace | item: Any, new\_priority\_value: int | None | If the item given exists in the priority queue, this function will change the priority of that item to the new priority value we have been given. |
| is\_empty | None | bool | This function will check if the priority queue is empty. |
| exists | item: Any | bool | This will check if the item given exists in the self.heap list. |
| \_\_init\_\_ | None | None | Initializes the PriorityQueue class. |
| dequeue | None | Any | This function will remove the item which has the highest priority from the priority queue as long as the queue is not empty. |
| peek | None | Any | This function will return the item with the highest priority in the priority queue. |
| enqueue | item: Any, priority: int | None | Will add an item to the queue with the given priority. |

### Queue Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_str\_\_ | None | Str | Whenever we try to print out the instance of the Queue class it will instead print the self.heap list. |
| \_\_iter\_\_ | None | Iterator | Makes it so that when we loop over the instance of the Queue class it will actually loop over the self.heap list. |
| is\_empty | None | bool | This function will check if the queue is empty. |
| exists | item: Any | bool | This function will check if the item given exists in the self.heap list. |
| remove\_empty\_values | None | None | This function will go through the self.heap list and find any elements which are the None value, and it will remove them from the list. |
| \_\_init\_\_ | None | None | Initializes the Queue class. |
| dequeue | None | Any or None | This function will remove the first element in the queue as long as the queue is not empty. |
| peek | None | ANy | This function will return the first item in the self.heap list. |
| enqueue | item: Any | None | This function will enqueue an item into the queue. |

### RandomMarkedMaze Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| create\_random\_marked\_maze | None | None | This algorithm will go through each RectNode in self.rect\_array\_obj.array and randomly choose whether the RectNode should be marked, if we decided that the RectNode should be marked we will set the RectNode's marked attribute to True. We will also animate this RectNode using the add\_coords\_to\_animation\_dict method in self.animation\_manager. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArrayObj, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the RandomMarkedMaze class. |

### RandomWeightedMaze Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| create\_random\_weighted\_maze | None | None | This algorithm will go through each RectNode in self.rect\_array\_obj.array and randomly choose whether the RectNode should be weighted, if we decided that the RectNode should be weighted we will choose a random weight for it and set the RectNode's is\_user\_weight attribute to True and the weight attribute to the weight we have randomly generated. We will also animate this RectNode using the add\_coords\_to\_animation\_dict method in self.animation\_manager. |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArray, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the RandomWeightedMaze class. |

### Rect Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |

### RectArray Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| reset\_non\_user\_weights | None | None | This function will go through each RectNode in the grid and will set the weight attribute of the RectNode to 1 if the is\_user\_weight attribute is set to False. |
| gen\_rect\_array\_with\_adjacent\_nodes | None | None | This function will go through each RectNode in self.array and calculate the coordinates of its adjacent nodes (i.e. if any node exists above, below, to the right or to the left of the current node and that node is not marked, then that node will be considered an adjacent node to our current node). It will then fill in the adjacent\_nodes attribute of the RectNode with the coordinates of its adjacent nodes if they exist. |
| get\_valid\_adjacent\_nodes | coords: List | List | This function will find the RectNode at the coordinates given in self.array, and then go through the adjacent\_nodes attribute of the RectNode and return a new array only containing the coordinates of the adjacent nodes (this list will not contain any 'None' values which may exist in the adjacent\_nodes attribute). |
| reset\_rect\_array | None | None | This function will call the gen\_rect\_array() method. |
| get\_weight\_at\_node | coord: List | int | This function will return the value of the weight attribute of the RectNode at the given coordinates from self.array. |
| reset\_rect\_array\_adjacent\_nodes | None | None | This function will go through each RectNode in self.array and set the adjacent\_nodes attribute of each RectNode to the list [None, None, None, None]. |
| set\_weight\_at\_node | coord: List, weight: bool | None | This function will get the RectNode at the coordinates given from self.array and set the weight attribute of the RectNode to the weight given if the is\_user\_weight attribute is set to False. |
| \_\_init\_\_ | screen\_manager: ScreenManager | None | Initialises the RectArray class. |
| get\_start\_and\_end\_node\_coords | None | Tuple | This function will go through self.array and find the coordinates of the start the end nodes and then return those coordinates. |
| gen\_rect\_array | None | None | This function will make self.array into 2D array which represents the game's grid. Each row in self.array will contain multiple RectNode instances (each row should have enough RectNode instances to match the number of columns in the grid) which will represent each cell on the grid. The top left most node should be set as the start node and the bottom right most node should be set as the end node. |

### RectNode Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_init\_\_ | rect: pygame.Rect, coords: List, is\_start\_node: bool, is\_end\_node: bool, is\_user\_weight: bool, weight: int, marked: bool, adjacent\_nodes: List | None | Initialises the RectNode class. |

### RecursiveDivisionMaze Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| run\_recursive\_division | None | None | This function is called to run recursive division, it will reset the self.maze stack using the reset\_maze method as well as making the self.empty\_nodes\_x and self.empty\_nodes\_y attributes both point to empty Stacks which are have the same size as the number of nodes in the grid (this can be calculated by multiplying the num\_of\_rows and num\_of\_columns attributes found in self.screen\_manager). This function will then call the recursive\_division function which will recursively call itself until it has finished generated the maze (it will push all the coordinates in the maze to the self.maze stack). |
| \_\_init\_\_ | screen\_manager: ScreenManager, rect\_array\_obj: RectArrayObj, color\_manager: ColorManager, animation\_manager: AnimationManager | None | Initializes the RecursiveDivisionMaze class. |
| recursive\_division | start\_x: int, start\_y: int, end\_x: int, end\_y: int, skew: RecursiveDivisionSkew | None | This function recursively calls itself to generate the Recursive Division Maze, and it will push the coordinates of the marked nodes in the maze to the self.maze stack. |

### ScreenManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| increment\_resolution\_divider | None | None | This function will increment the value of self.resolution\_divider as long as the value of self.resolution\_divider is less than 8. |
| set\_resolution\_divider | resolution\_divider: int | None | Setter to set the value of the self.resolution\_divider attribute. |
| decrement\_resolution\_divider | None | None | This function will decrement the value of self.resolution\_divider as long as the value of self.resolution\_divider is greater than 1. |
| \_\_init\_\_ | screen: pygame.Surface, screen\_width: int, screen\_height: int, grid\_width: int, grid\_height: int, resolution\_divider: int | None | Initialises the ScreenManager class. |

### Server Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| client\_server\_loop | None | None | This function will be run in a separate thread. It will run a loop where it will continuously add clients who are trying to join the server to the connected\_clients\_dict dictionary as long as the server\_running attribute is set to True. Once it has accepted a client it will get a list containing information about the grid from the get\_board\_info method in self.grid. It will append the pathfinding\_algorithm\_speed and recursive\_division\_speed attributes to this list and send it to the client. In addition to this, this function will get information about the current theme name using the current\_theme\_name attribute in self.color\_manager as well as the dictionary containing the colours the theme uses from the get\_theme\_colors\_dict method in self.color\_manager, it will then send this information to the client. After this, the function will run the handle\_client method in a separate thread and pass in information about the client's address and socket. |
| shutdown | None | None | This function will shut down the server by doing the following: 1) Running the kick\_out\_clients method. 2) Closing the socket running on the server\_socket attribute 3) Set the connected\_to\_clients\_dict dictionary to an empty dictionary. 4) Set the server\_running attribute to False 5) Set the server\_socket attribute to None. |
| run\_server | ip\_address: Str, port: int | None | If the server\_running variable is set to False, this function will set the server\_running variable to True and create a new TCP socket on the server\_socket attribute which will be bound to the IP Address and Port Number we have been given. We will then run the client\_server\_loop method in a separate thread. |
| get\_number\_of\_currently\_connected\_clients | None | int | Returns the number of key-value pairs in the connected\_clients\_dict dictionary. |
| set\_pathfinding\_algorithm\_speed\_and\_recursive\_division\_speed | pathfinding\_algorithm\_speed: int, recursive\_division\_speed: int | None | This function is a setter for both the pathfinding\_algorithm\_speed and recursive\_division\_speed attributes. |
| \_\_init\_\_ | grid: Grid, color\_manager: ColorManager | None | Initalizes the Server class. |
| kick\_out\_clients | None | None | This function will go through each client in the connected\_clients\_dict dictionary and send them a dictionary whose key will be NetworkingEventTypes.DISCONNECT\_FROM\_SERVER and this key will have a value of None (this will tell all the clients to disconnect from the server). |
| handle\_client | client\_socket: socket.socket, client\_address: Str | None | This function will be run in a separate thread. It will run a loop which will continuously check for new messages from the client\_socket as long as the server\_running attribute is set to True. When it receives a message from the client it will turn the json string into a dictionary and check if the event type of the dictionary is NetworkingEventTypes.DISCONNECT\_FROM\_SERVER in which case it will remove the client from the connected\_clients\_dict dictionary and exit the function. If the event is not NetworkingEventTypes.DISCONNECT\_FROM\_SERVER the server will send the dictionary to all the other clients in the connected\_clients\_dict dictionary. |

### SettingsWindow Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| handle\_settings\_window\_ui\_selection\_list\_new\_selection | event: pygame.event | None | Handles the pygame\_gui.UI\_SELECTION\_LIST\_NEW\_SELECTION event for SettingsWindow. |
| handle\_settings\_window\_ui\_text\_entry\_finished\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_TEXT\_ENTRY\_FINISHED event for SettingsWindow. |
| handle\_settings\_window\_ui\_corner\_roundness\_changed | None | None | This function will perform the relevant operations to ensure that the corner roundness of the UI Elements is the same as the ui\_corner\_roundness\_value attribute. |
| handle\_settings\_window\_ui\_drop\_down\_menu\_changed\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_DROP\_DOWN\_MENU\_CHANGED event for SettingsWindow. |
| handle\_settings\_window\_border\_width\_changed | None | None | This function will set the changed\_ui\_border\_width attribute to True, and it will perform the relevant operations to ensure that the border width of the UI Elements is the same as the ui\_border\_width\_value attribute. |
| handle\_settings\_window\_ui\_horizontal\_slider\_moved\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_HORIZONTAL\_SLIDER\_MOVED event for SettingsWindow. |
| handle\_settings\_window\_ui\_file\_dialog\_path\_picked\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_FILE\_DIALOG\_PATH\_PICKED event for SettingsWindow. |
| should\_build\_tutorial\_window\_on\_startup | None | bool | This function will return the result of performing the logical negation operation on the boolean stored as the value of the 'disabled\_tutorial\_on\_startup' key in the user\_settings\_json\_data dictionary. |
| \_\_init\_\_ | manager: pygame\_gui.ui\_manager.UIManager, theme\_manager: pygame\_gui.core.interfaces.appearance\_theme\_interface.IUIAppearanceThemeInterface, theme\_json\_data: Dict, grid: Grid, font\_manager: FontManager | None | Initializes the SettingsWindow class. |
| shutdown | None | None | Sets the window\_running attribute to False. |
| build\_settings\_window | None | None | It will first create a window and then create the widgets for SettingsWindowStages.SETTINGS\_WINDOW\_SETTINGS\_SCREEN. After this it will run the handle\_settings\_window\_border\_width\_changed method. |
| set\_disable\_tutorial\_window\_on\_startup | option: bool | None | This function will set the value of the 'disable\_tutorial\_on\_startup' key in the user\_settings\_json\_data dictionary to the boolean value option given. After this it will run the save\_user\_settings method. |
| clean\_custom\_font\_creation\_window | None | None | Removes the widgets for SettingsWindowStages.SETTINGS\_WINDOW\_CUSTOM\_FONT\_CREATION\_SCREEN. |
| handle\_settings\_window\_ui\_button\_pressed\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_BUTTON\_PRESSED event for SettingsWindow. |
| clean\_custom\_font\_deletion\_window | None | None | Removes the widgets for SettingsWindowStages.SETTINGS\_WINDOW\_CUSTOM\_FONT\_DELETION\_SCREEN. |
| build\_custom\_font\_deletion\_window | None | None | Creates the widgets for SettingsWindowStages.SETTINGS\_WINDOW\_CUSTOM\_FONT\_DELETION\_SCREEN. |
| save\_user\_settings | None | None | This function will turn the dictionary stored in the user\_settings\_json\_data attribute into a json string, and we will overwrite the file at data/settings/user\_settings.json with this json string. |
| clean\_settings\_window | None | None | Removes the widgets for SettingsWindowStages.SETTINGS\_WINDOW\_SETTINGS\_SCREEN. |
| build\_custom\_font\_creation\_window | None | None | Creates the widgets for SettingsWindowStages.SETTINGS\_WINDOW\_CUSTOM\_FONT\_CREATION\_SCREEN. |
| load\_user\_settings | None | None | This function will first open the data/settings/user\_settings.json file, and it will read all the json data in the file and convert it into a dictionary which will be stored in the user\_settings\_json\_data attribute. We will then perform the relevant operations required to apply the settings in the user\_settings\_json\_data dictionary to the UI and the grid (this consists of things such as the corner roundness of UI Elements, the shape of UI Elements, the border width of the UI Elements, the thickness of the lines on the grid, the font size of normal UI Elements, the font size of titles in the UI and the font we should be using in the UI. |
| handle\_settings\_window\_ui\_selection\_list\_dropped\_selection | event: pygame.event | None | Handles the pygame\_gui.UI\_SELECTION\_LIST\_DROPPED\_SELECTION event for SettingsWindow. |

### Stack Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| get\_size | None | int | This function will return the number of elements in the stack. |
| \_\_str\_\_ | None | Str | Whenever we try to print out the instance of the Stack class it will instead print the self.stack list. |
| merge | stack1: Stack, stack2: Stack | None | This function will run the remove\_empty\_values method on the stack1 and stack2 stacks we have been given. It will then combine these two stacks and make this combined stack the value of the self.stack attribute. |
| \_\_iter\_\_ | None | Iterator | Makes it so that when we loop over the instance of the Stack class it will actually loop over the self.stack list. |
| gen\_copy\_without\_empty\_values | None | List | This function will create a copy of the self.stack list, and remove any None values in the list. It will then return this list. |
| exists | item: Any | bool | This function will check if the item given exists in the self.stack list. |
| pop | show\_errors: bool | None or -1 | If the stack is not empty, this function will remove the element in the stack at the index of the pointer attribute, it will then decrement the pointer attribute. If the stack is empty and the show\_errors variable given is equal to True we will print out an error message. |
| remove\_empty\_values | None | None | This function will go through the self.stack list and find any elements which are the None value, and it will remove them from the list. |
| \_\_init\_\_ | size: int | None | Initializes the Stack class. |
| peek | show\_errors: bool | Any or -1 | If the stack is not empty, we will print out the element in the stack at the index of the pointer attribute. If the stack is empty and the show\_errors variable is set to True we will print out an error message. |
| push | value: Any, show\_errors: bool | None or -1 | If the stack is not full, this function will add an element to the stack and also increment the pointer attribute. If the stack is full and the show\_errors variable given is equal to True we will print out an error message. |
| to\_list | None | List | This function will run the remove\_empty\_values method and then return the self.stack list. |
| reverse | None | None | This function will run the remove\_empty\_values method and then reverse the self.stack list. |

### ThemeWindow Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| build\_custom\_theme\_creation\_welcome\_screen | None | None | It will first create a window and then create the widgets for ThemeWindowStages.CUSTOM\_THEME\_CREATION\_WELCOME\_SCREEN. |
| handle\_theme\_window\_ui\_drop\_down\_menu\_changed\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_DROP\_DOWN\_MENU\_CHANGED event for ThemeWindow. |
| clean\_custom\_theme\_editing\_welcome\_screen | None | None | Removes widgets for ThemeWindowStages.CUSTOM\_THEME\_EDITING\_WELCOME\_SCREEN. |
| build\_custom\_theme\_creation\_selection\_screen | None | None | Creates the widgets for ThemeWindowStages.CUSTOM\_THEME\_CREATION\_COLOR\_SELECTION\_SCREEN. |
| build\_custom\_theme\_creation\_finish\_screen | None | None | Creates the widgets for ThemeWindowStages.CUSTOM\_THEME\_CREATION\_FINISH\_SCREEN. |
| \_\_init\_\_ | manager: pygame\_gui.ui\_manager.UIManager, color\_manager: ColorManager, font\_manager: FontManager | None | Initializes the ThemeWindow class. |
| handle\_theme\_window\_ui\_color\_picker\_color\_picked\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_COLOUR\_PICKER\_COLOUR\_PICKED event for ThemeWindow. |
| build\_custom\_theme\_editing\_welcome\_screen | None | None | Creates widgets for ThemeWindowStages.CUSTOM\_THEME\_EDITING\_WELCOME\_SCREEN. |
| handle\_theme\_window\_ui\_selection\_list\_new\_selection | event: pygame.event | None | Handles the pygame\_gui.UI\_SELECTION\_LIST\_NEW\_SELECTION event for ThemeWindow. |
| handle\_theme\_window\_ui\_selection\_list\_dropped\_selection | event: pygame.event | None | Handles the pygame\_gui.UI\_SELECTION\_LIST\_DROPPED\_SELECTION event for ThemeWindow. |
| clean\_custom\_theme\_delete\_screen | None | None | Removes widgets for ThemeWindowStages.CUSTOM\_THEME\_DELETE\_SCREEN. |
| clean\_custom\_theme\_editing\_finish\_screen | None | None | Removes widgets for ThemeWindowStages.CUSTOM\_THEME\_EDITING\_FINISH\_SCREEN. |
| handle\_theme\_window\_ui\_button\_pressed\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_BUTTON\_PRESSED event for ThemeWindow. |
| build\_custom\_theme\_editing\_finish\_screen | None | None | Creates widgets for ThemeWindowStages.CUSTOM\_THEME\_EDITING\_FINISH\_SCREEN. |
| build\_custom\_theme\_editing\_color\_selection\_screen | None | None | Creates widgets for ThemeWindowStages.CUSTOM\_THEME\_EDITING\_COLOR\_SELECTION\_SCREEN. |
| clean\_custom\_theme\_editing\_color\_selection\_screen | None | None | Removes widgets for ThemeWindowStages.CUSTOM\_THEME\_EDITING\_COLOR\_SELECTION\_SCREEN. |
| handle\_theme\_window\_ui\_text\_entry\_finished\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_TEXT\_ENTRY\_FINISHED event for the ThemeWindow. |
| build\_custom\_theme\_delete\_screen | None | None | Creates widgets for ThemeWindowStages.CUSTOM\_THEME\_DELETE\_SCREEN. |
| clean\_custom\_theme\_creation\_finish\_screen | None | None | Removes the widgets for ThemeWindowStages.CUSTOM\_THEME\_CREATION\_FINISH\_SCREEN. |
| clean\_custom\_theme\_creation\_welcome\_screen | None | None | Removes the widgets for ThemeWindowStages.CUSTOM\_THEME\_CREATION\_WELCOME\_SCREEN. |
| shutdown | None | None | Sets the window\_running attribute to False. |
| clean\_custom\_theme\_creation\_selection\_screen | None | None | Removes the widgets for ThemeWindowStages.CUSTOM\_THEME\_CREATION\_COLOR\_SELECTION\_SCREEN. |

### TutorialWindow Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| build\_pathfinding\_algorithm\_info\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_PATHFINDING\_ALGORITHM\_INFO. |
| clean\_tutorial\_welcome\_screen | None | None | Removes the widgets TutorialWindowStages.TUTORIAL\_WELCOME\_SCREEN. |
| clean\_pathfinding\_algorithm\_info\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_PATHFINDING\_ALGORITHM\_INFO. |
| build\_project\_description\_screen | None | None | Creates widgets for TutorialWindowStages.TUTORIAL\_PROJECT\_DESCRIPTION. |
| build\_fonts\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_FONTS. |
| clean\_end\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_END\_SCREEN. |
| clean\_pathfinding\_algorithm\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_PATHFINDING\_ALGORITHM. |
| build\_end\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_END\_SCREEN. |
| build\_maze\_generation\_algorithm\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_MAZE\_GENERATION\_ALGORITHM. |
| build\_theming\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_THEMING. |
| clean\_settings\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_SETTINGS. |
| build\_clearing\_nodes\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_CLEARING\_NODES. |
| fill\_image\_with\_color\_and\_set\_shadow\_to\_border\_color | key: str, color: tuple, shadow\_colour: tuple | None | This function will go through each pixel in the image specified by the key (we will get the image data by accessing with the key given in ui\_tutorial\_image\_surfaces\_dict dictionary) and it will check the value of the alpha channel of the pixel. If the pixel is transparent then we will continue to the next iteration otherwise we will check if the rgb values of the pixel are the same as those specified in the shadow\_colour given. If it is, we will make the rgb value of this pixel be the same as the UI\_BORDER\_COLOR (we can get this by accessing the UI\_BORDER\_COLOR method in self.color\_manager). If the rgb values are not the same as the shadow\_colour we will instead set the rgb values of the pixel to be the same as the color given. |
| load\_tutorial\_assets | None | None | This function will go through each image in the data/tutorial\_assets directory and remove the file extension from the image file's name. It will then add this image file name (without the file extension) as a key for the ui\_tutorial\_images\_surfaces\_dict dictionary with the value for the key being the loaded image file (this should be a pointer to or a way for us to access the image file in memory so that it is ready to draw onto the screen). |
| \_\_init\_\_ | manager: pygame\_gui.ui\_manager.UIManager, color\_manager: ColorManager, font\_manager: FontManager, settings\_window: SettingsWindow | None | Initializes the TutorialWindow class. |
| build\_networking\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_NETWORKING. |
| clean\_networking\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_NETWORKING. |
| build\_pathfinding\_algorithm\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_PATHFINDING\_ALGORITHM. |
| build\_marked\_and\_weighted\_nodes\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_MARKED\_AND\_WEIGHTED\_NODES. |
| build\_tutorial\_welcome\_screen | None | None | It will first create a window and then create the widgets for TutorialWindowStages.TUTORIAL\_WELCOME\_SCREEN. |
| clean\_marked\_and\_weighted\_nodes\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_MARKED\_AND\_WEIGHTED\_NODES. |
| clean\_clearing\_nodes\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_CLEARING\_NODES. |
| clean\_maze\_generation\_algorithm\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_MAZE\_GENERATION\_ALGORITHM. |
| clean\_theming\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_THEMING. |
| handle\_tutorial\_window\_ui\_button\_pressed | event: pygame.event | None | Handles the pygame\_gui.UI\_NETWORKING\_MANAGER\_UI\_BUTTON\_PRESSED event for TutorialWindow. |
| clean\_fonts\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_FONTS. |
| clean\_maze\_generation\_algorithm\_info\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_MAZE\_GENERATION\_ALGORITHM\_INFO. |
| clean\_project\_description\_screen | None | None | Removes the widgets for TutorialWindowStages.TUTORIAL\_PROJECT\_DESCRIPTION. |
| build\_settings\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_SETTINGS. |
| shutdown | None | None | Sets the window\_running attribute to False. |
| build\_maze\_generation\_algorithm\_info\_screen | None | None | Creates the widgets for TutorialWindowStages.TUTORIAL\_MAZE\_GENERATION\_ALGORITHM\_INFO. |

### UIAnimationNode Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| \_\_init\_\_ | initial\_colour: pygame.Color, final\_colour: pygame.Color | None | Initializes the UIAnimationNode class. |

### UINetworkingManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | Return Value | Description |
| show\_server\_info | None | None | Creates a message window which will tell the user the following information about the game server running on their local machine. - IP Address of the server. - Port Number the server is listening on. - How many clients are connected to the server. |
| create\_server | None | None | Creates a message window telling the user that they have successfully created a server running on their local machine. This message window will also tell the user information about the server (i.e. the IP Address and Port Number the server is running on). |
| disconnect\_from\_server | None | None | This function will set the created\_server and connected\_to\_server attributes to False. It will then create a network event using the create\_network\_event method in self.client with the event NetworkingEventTypes.DISCONNECT\_FROM\_SERVER. After this, the function will create a message window telling the user that they have successfully disconnected from the server. |
| server\_connection\_has\_broken | None | None | This function will set both the created\_server and connected\_to\_server attributes to False. It will then check if the shutdown\_server attribute is set to True, if it is we will set to it to False, otherwise we will create a message window telling the user that the server they were connected to has been shut down by the owner. |
| check\_is\_valid\_port | port\_number: int | bool | This function will check if the port\_number given is in the range 2000 to 9999. |
| shutdown\_networking\_window | None | None | Sets the window\_running attribute to False. |
| build\_networking\_connect\_to\_server\_screen | None | None | It will first create a window and then create the widgets for the screen to connect to server. |
| destroy\_server | None | None | This function is used to shut down a server and will do the following: - The shutdown\_server attribute is set to True - The created\_server attribute is set to False - The connected\_to\_server attribute is set to False - The shutdown() method is called from self.server - A message window is created telling the user that the server has been shutdown successfully. |
| check\_is\_valid\_ip\_address | ip\_address: Str | bool | This function will take in a string and check if it is the valid format to be considered an IP Address. |
| \_\_init\_\_ | manager: pygame\_gui.ui\_manager.UIManager, client: Client, server: Server, font\_manager: FontManager | None | Initializes the UINetworkingManager class. |
| clean\_networking\_connect\_server\_screen | None | None | Removes the widgets for the screen to connect to the server. |
| handle\_ui\_networking\_manager\_ui\_button\_pressed\_event | event: pygame.event | None | Handles the pygame\_gui.UI\_BUTTON\_PRESSED event for UINetworkingManager. |