# Final Project Write-UP

Group 99
Sade Yokubova and Matei Obrocea

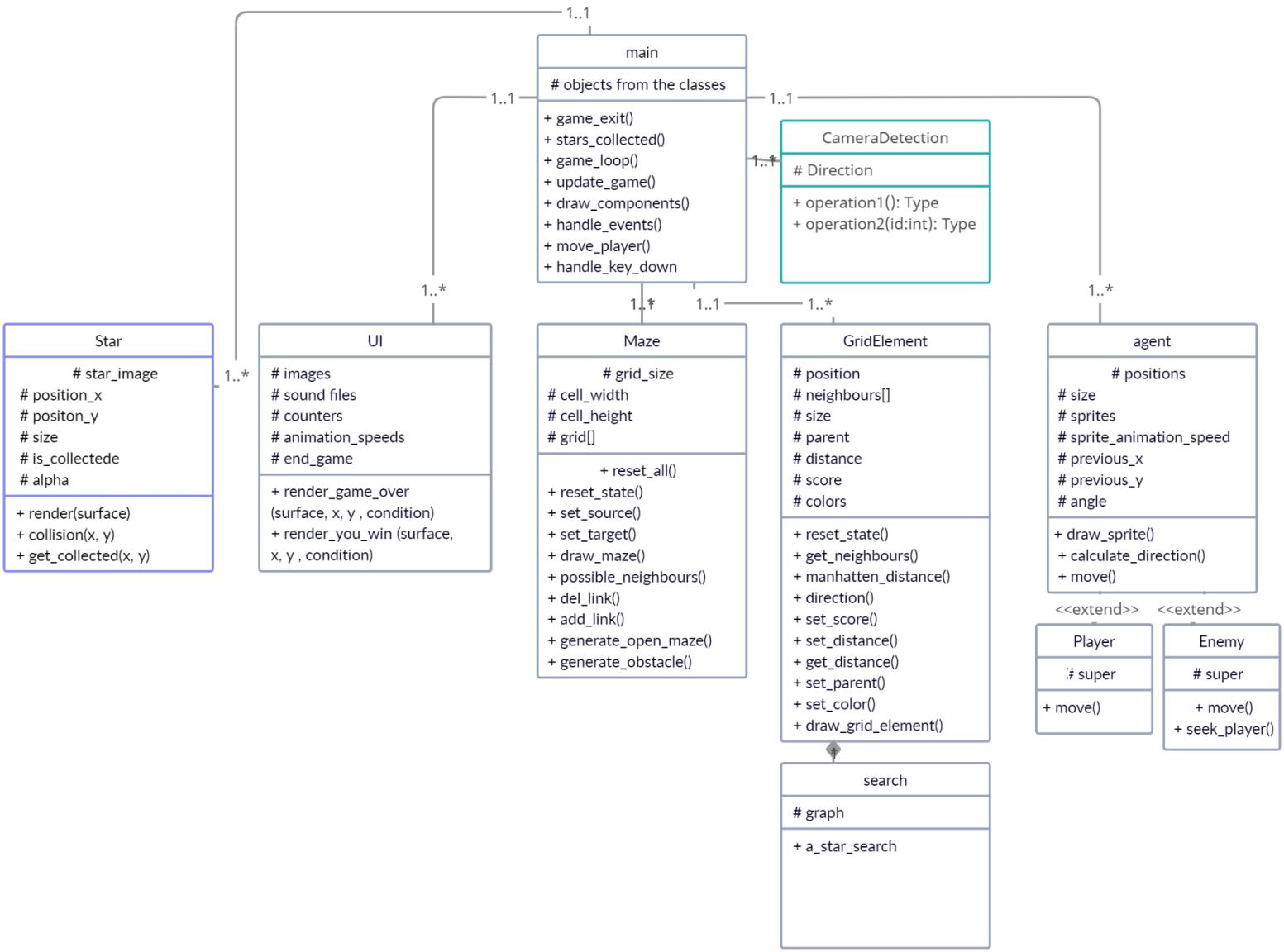
In this project, we developed an arcade game using the pygame library, which has a standard maze, an enemy, and a player. Our main character is a raven whose goal is to collect all the stars on the display in order to win. But the raven will be hindered by his enemy, in our case it is a cat.

The non-standard input, together with the library, are accounted for by the Open-Cv library, which we used to control the raven. The program detects 2 colors and computes the angle between them, using trigonometry. These colors will be painted on an external tool, namely a 3D printed arrow, which will indicate the direction of the movement as follows: if the arrow points up, the raven moves up, if it points right, the raven moves to the right, and so on.

The game is based on the week 6 assignment of this class, namely the search assignment. On a similar grid, the program computes the path between the source (the cat) and the target (the raven) using the a* search algorithm, which the cat will follow.

Either when the player has collected all stars or when the cat catches the raven, the program displays corresponding messages and ends.

The diagram below shows the structure of the software:


1..1
main
# objects from the classes
+ game_exit()
+ stars_collected()
+ game_loop()
+ update_game()
+ draw_components()
+ handle_events()
+ move_player()
+ handle_key_down
1..1
1..1
CameraDetection
# Direction
+ operation1(): Type
+ operation2(id:int): Type
1..*
1..*
1..1
1..1
1..*
1..*
Star
# star_image
# position_x
# positon_y
# size
# is_collectede
# alpha
+ render(surface)
+ collision(x, y)
+ get_collected(x, y)
UI
# images
# sound files
# counters
# animation_speeds
# end_game
+ render_game_over
(surface, x, y , condition)
+ render_you_win (surface,
x, y , condition)
Maze
# grid_size
# cell_width
# cell_height
# grid[]
+ reset_all()
+ reset_state()
+ set_source()
+ set_target()
+ draw_maze()
+ possible_neighbours()
+ del_link()
+ add_link()
+ generate_open_maze()
+ generate_obstacle()
GridElement
# position
# neighbours[]
# size
# parent
# distance
# score
# colors
+ reset_state()
+ get_neighbours()
+ manhatten_distance()
+ direction()
+ set_score()
+ set_distance()
+ get_distance()
+ set_parent()
+ set_color()
+ draw_grid_element()
search
# graph
+ a_star_search
agent
# positions
# size
# sprites
# sprite_animation_speed
# previous_x
# previous_y
# angle
+ draw_sprite()
+ calculate_direction()
+ move()
<<extend>>
<<extend>>
Player
# super
+ move()
Enemy
# super
+ move()
+ seek_player()