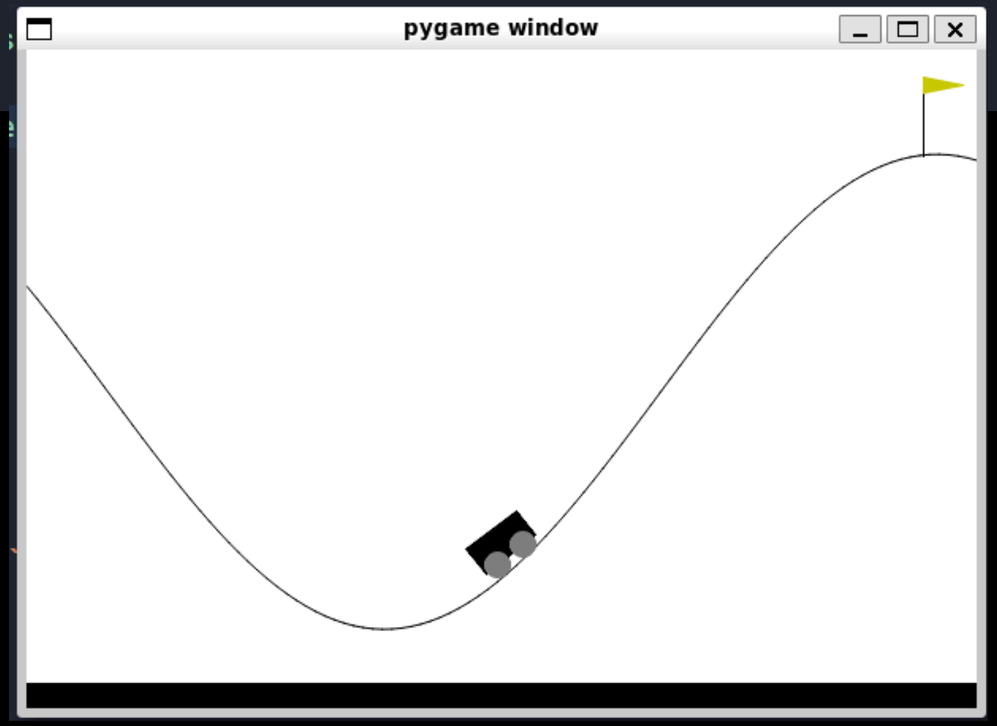
OOP GROUP PROJECT
TEAM 12

D E M O

B123040045 林柏儒
B123040048 吳紹彰
B123040053 張承勛

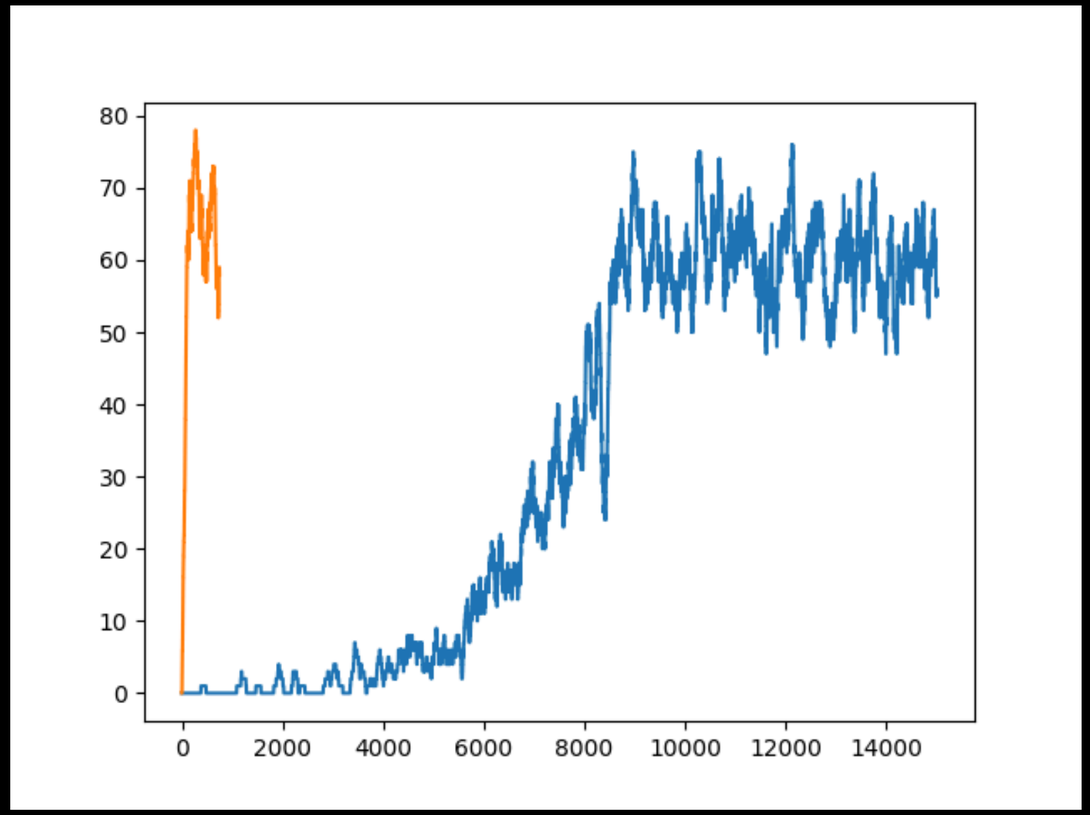
Part 1

```
(.venv) jonathanwu@Jonathan:~/NSYSU/OOP-Group-Project$ cd part1
(.venv) jonathanwu@Jonathan:~/NSYSU/OOP-Group-Project/part1$ python mountain_car.py --train --episodes 5000
Training for 5000 episodes...
(.venv) jonathanwu@Jonathan:~/NSYSU/OOP-Group-Project/part1$ python mountain_car.py --render --episodes 10
Running evaluation for 10 episodes (render=True)
```

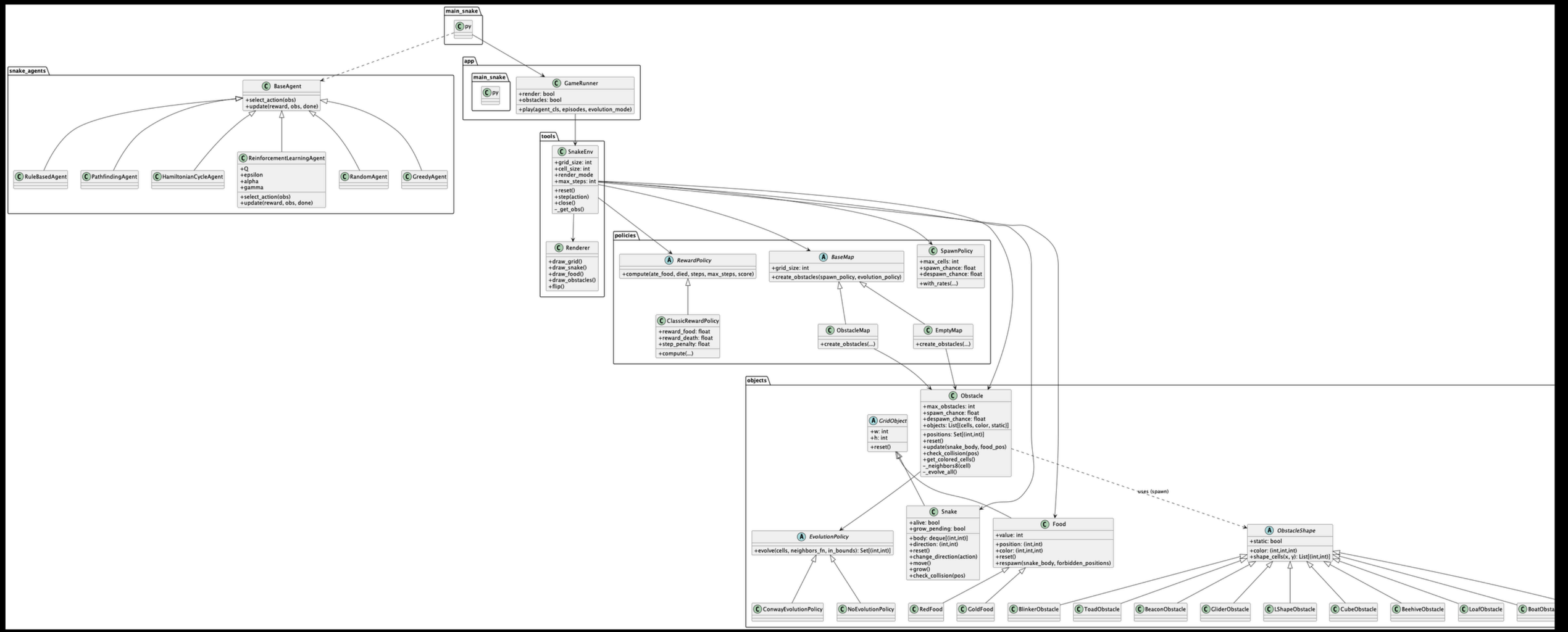


Part 2

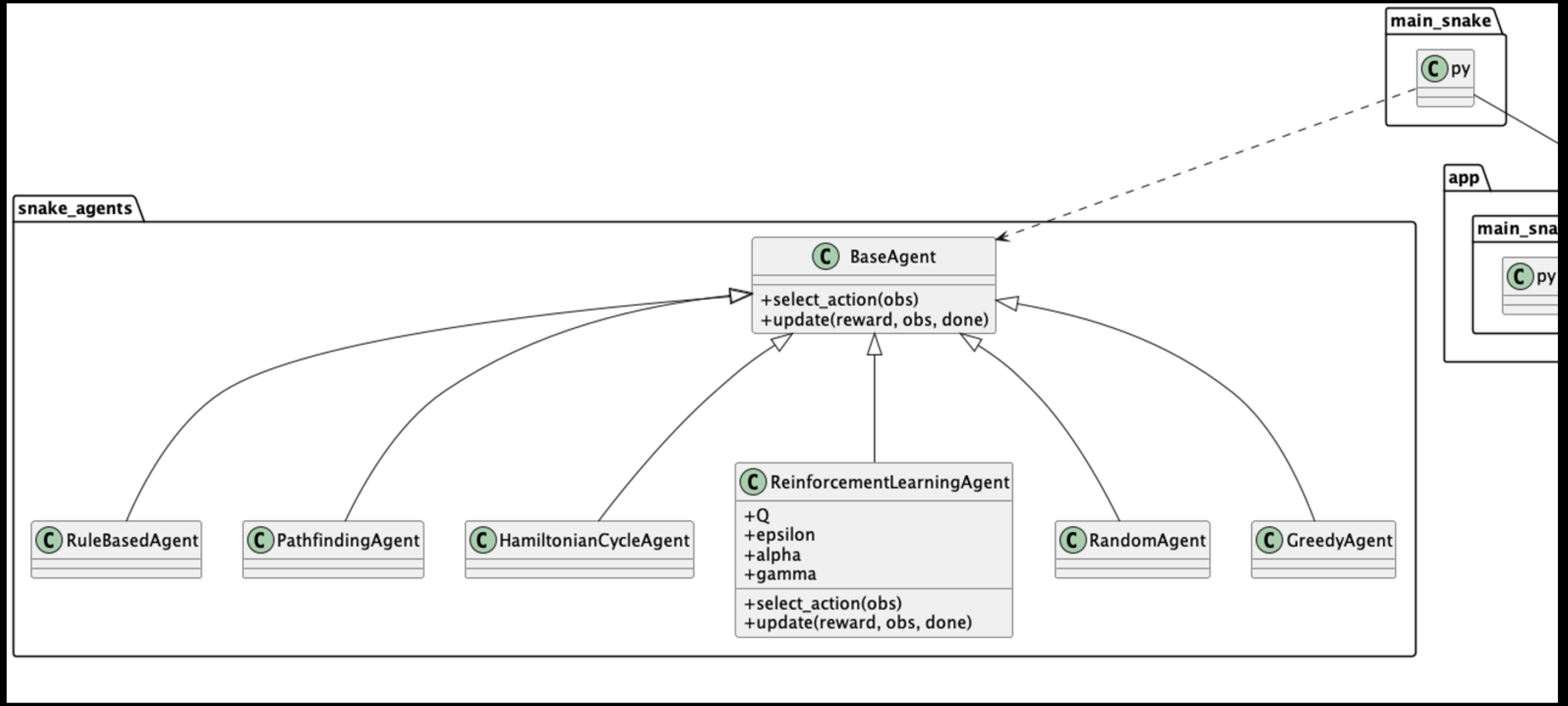
```
(.venv) jonathanwu@Jonathan:~/NSYSU/OOP-Group-Project/part2$ python3 frozen_lake.py
✅ Success Rate: 66.27% (497 / 750 episodes)
66.26666666666667
```



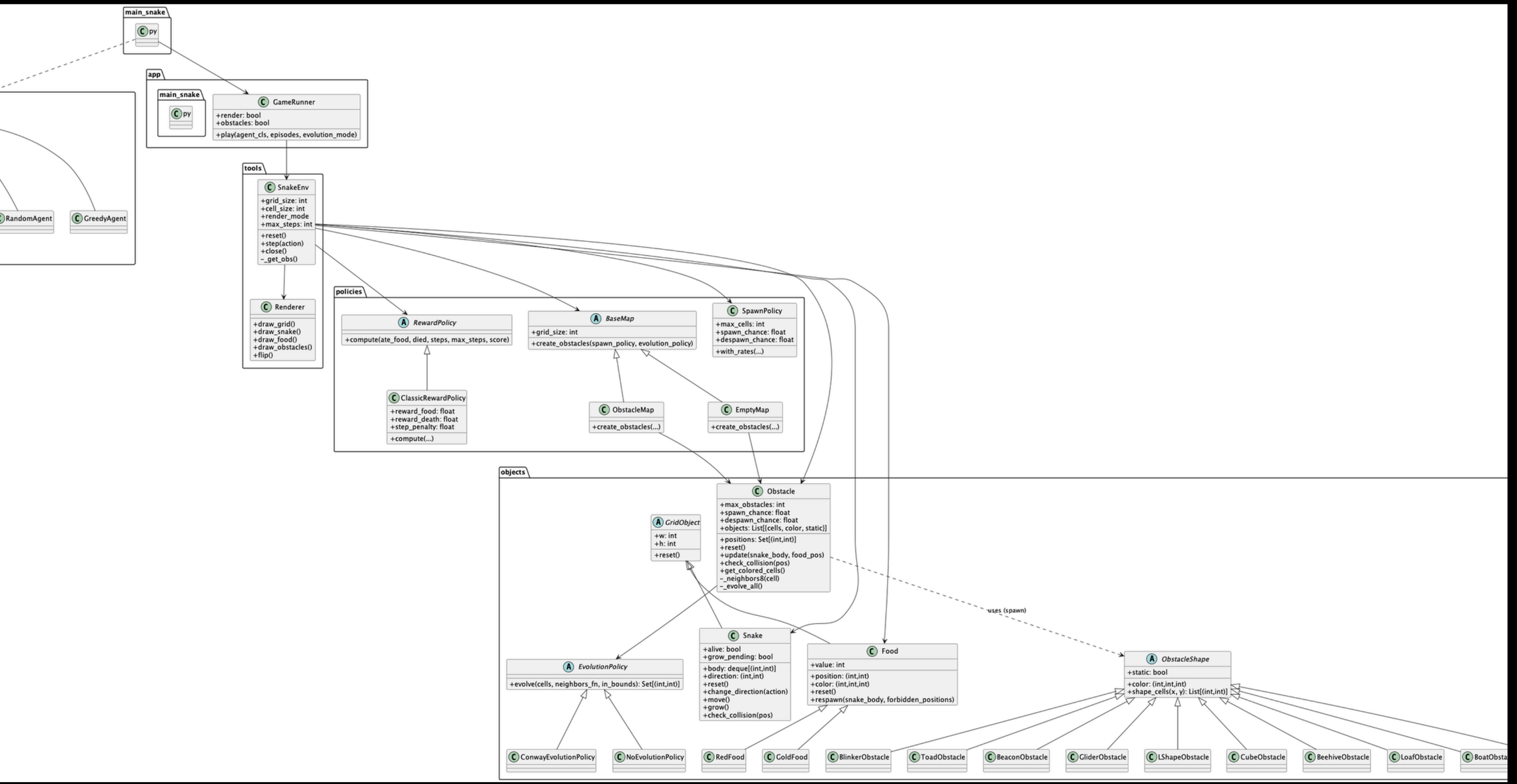
Part 3


main_snake
py
app
main_snake
py
GameRunner
+render: bool
+obstacles: bool
+play(agent_cls, episodes, evolution_mode)
snake_agents
BaseAgent
+select_action(obs)
+update(reward, obs, done)
RuleBasedAgent
PathfindingAgent
HamiltonianCycleAgent
ReinforcementLearningAgent
+Q
+epsilon
+alpha
+gamma
+select_action(obs)
+update(reward, obs, done)
RandomAgent
GreedyAgent
tools
SnakeEnv
+grid_size: int
+cell_size: int
+render_mode
+max_steps: int
+reset()
+step(action)
+close()
-_get_obs()
Renderer
+draw_grid()
+draw_snake()
+draw_food()
+draw_obstacles()
+flip()
policies
RewardPolicy
+compute(ate_food, died, steps, max_steps, score)
BaseMap
+grid_size: int
+create_obstacles(spawn_policy, evolution_policy)
SpawnPolicy
+max_cells: int
+spawn_chance: float
+despawn_chance: float
+with_rates(...)
ClassicRewardPolicy
+reward_food: float
+reward_death: float
+step_penalty: float
+compute(...)
ObstacleMap
+create_obstacles(...)
EmptyMap
+create_obstacles(...)
objects
Obstacle
+max_obstacles: int
+spawn_chance: float
+despawn_chance: float
+objects: List[(cells, color, static)]
+positions: Set[(int,int)]
+reset()
+update(snake_body, food_pos)
+check_collision(pos)
+get_colored_cells()
-_neighbors8(cell)
-_evolve_all()
GridObject
+w: int
+h: int
+reset()
uses (spawn)
Snake
+alive: bool
+grow_pending: bool
+body: deque[(int,int)]
+direction: (int,int)
+reset()
+change_direction(action)
+move()
+grow()
+check_collision(pos)
Food
+value: int
+position: (int,int)
+color: (int,int,int)
+reset()
+respawn(snake_body, forbidden_positions)
EvolutionPolicy
+evolve(cells, neighbors_fn, in_bounds): Set[(int,int)]
ObstacleShape
+static: bool
+color: (int,int,int)
+shape_cells(x, y): List[(int,int)]
ConwayEvolutionPolicy
NoEvolutionPolicy
RedFood
GoldFood
BlinkerObstacle
ToadObstacle
BeaconObstacle
GliderObstacle
LShapeObstacle
CubeObstacle
BeehiveObstacle
LoafObstacle

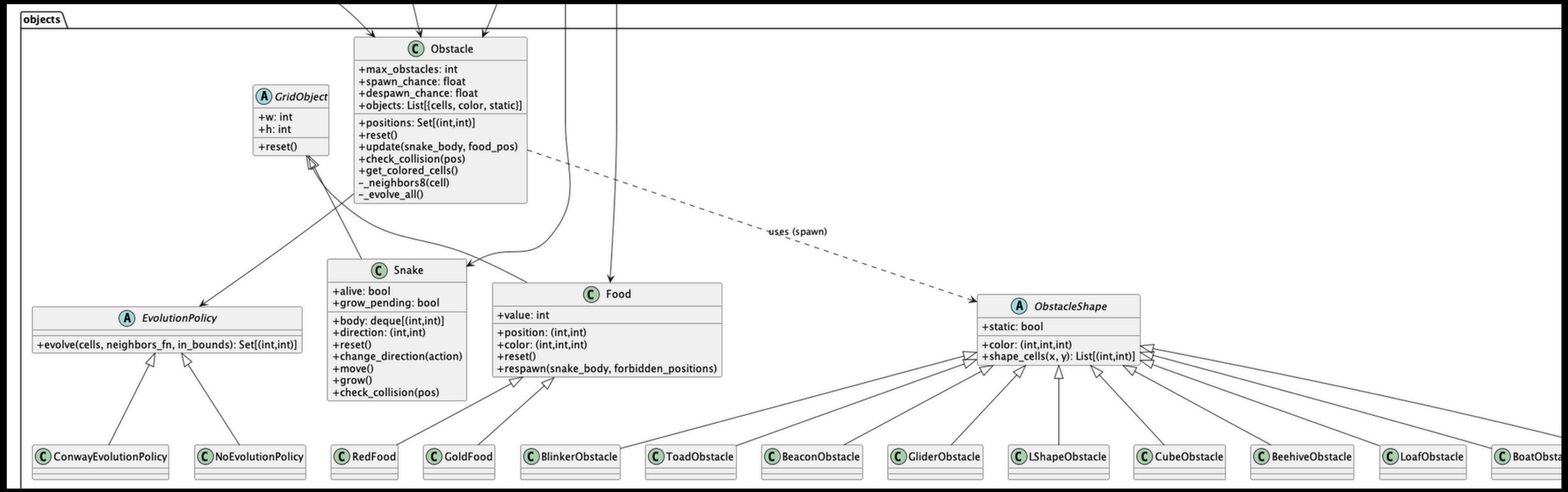
Part 3


main_snake
py
app
main_sna
py
snake_agents
BaseAgent
+select_action(obs)
+update(reward, obs, done)
RuleBasedAgent
PathfindingAgent
HamiltonianCycleAgent
ReinforcementLearningAgent
+Q
+epsilon
+alpha
+gamma
+select_action(obs)
+update(reward, obs, done)
RandomAgent
GreedyAgent

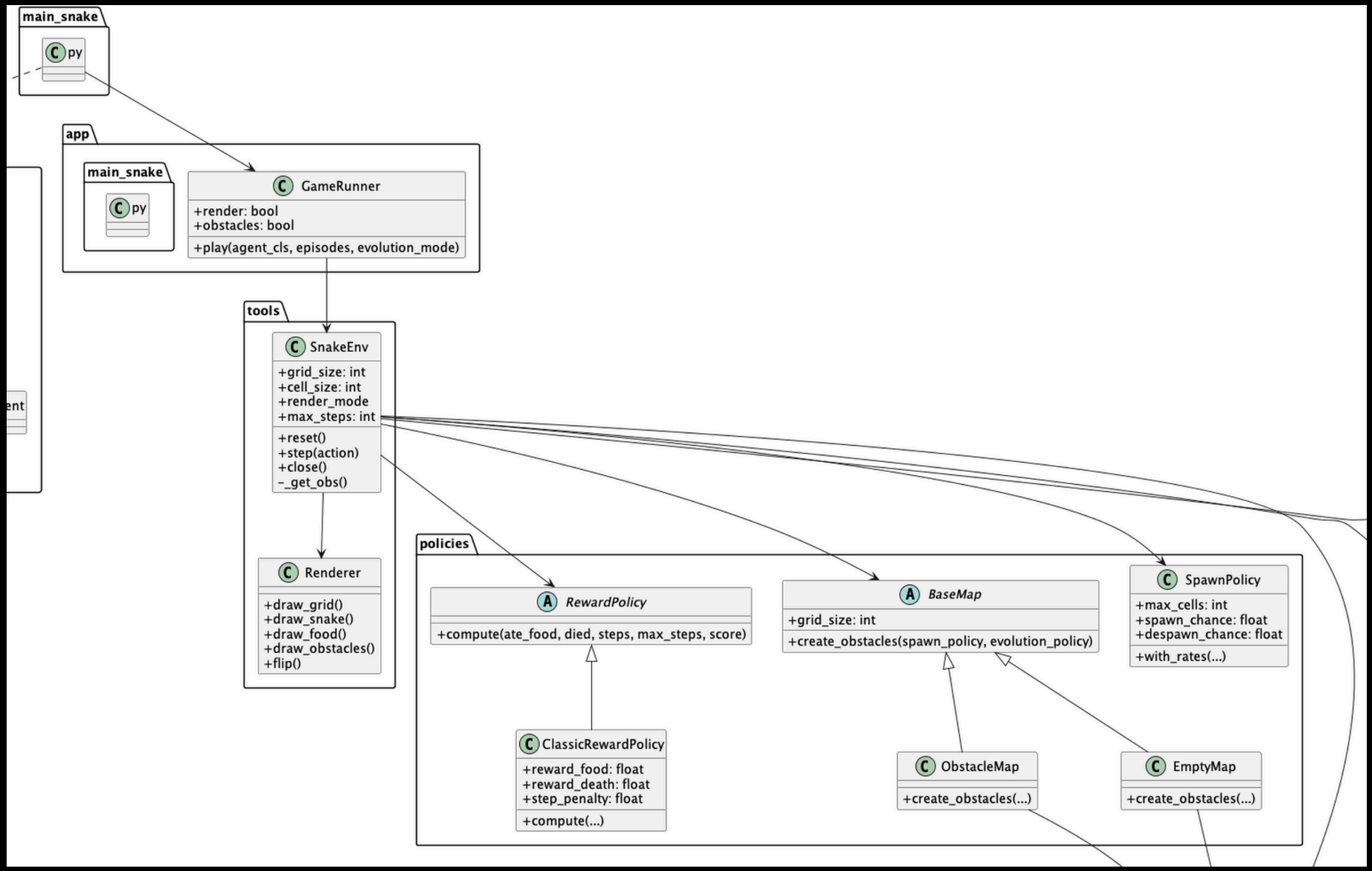
Part 3


main_snake
py
app
main_snake
py
GameRunner
+render: bool
+obstacles: bool
+play(agent_cls, episodes, evolution_mode)
RandomAgent
GreedyAgent
tools
SnakeEnv
+grid_size: int
+cell_size: int
+render_mode
+max_steps: int
+reset()
+step(action)
+close()
-_get_obs()
Renderer
+draw_grid()
+draw_snake()
+draw_food()
+draw_obstacles()
+flip()
policies
RewardPolicy
+compute(ate_food, died, steps, max_steps, score)
BaseMap
+grid_size: int
+create_obstacles(spawn_policy, evolution_policy)
SpawnPolicy
+max_cells: int
+spawn_chance: float
+despawn_chance: float
+with_rates(...)
ClassicRewardPolicy
+reward_food: float
+reward_death: float
+step_penalty: float
+compute(...)
ObstacleMap
+create_obstacles(...)
EmptyMap
+create_obstacles(...)
objects
Obstacle
+max_obstacles: int
+spawn_chance: float
+despawn_chance: float
+objects: List[(cells, color, static)]
+positions: Set[(int,int)]
+reset()
+update(snake_body, food_pos)
+check_collision(pos)
+get_colored_cells()
-_neighbors8(cell)
-_evolve_all()
GridObject
+w: int
+h: int
+reset()
uses (spawn)
Snake
+alive: bool
+grow_pending: bool
+body: deque[(int,int)]
+direction: (int,int)
+reset()
+change_direction(action)
+move()
+grow()
+check_collision(pos)
Food
+value: int
+position: (int,int)
+color: (int,int,int)
+reset()
+respawn(snake_body, forbidden_positions)
EvolutionPolicy
+evolve(cells, neighbors_fn, in_bounds): Set[(int,int)]
ObstacleShape
+static: bool
+color: (int,int,int)
+shape_cells(x, y): List[(int,int)]
ConwayEvolutionPolicy
NoEvolutionPolicy
RedFood
GoldFood
BlinkerObstacle
ToadObstacle
BeaconObstacle
GliderObstacle
LShapeObstacle
CubeObstacle
BeehiveObstacle
LoafObstacle
BoatObsta

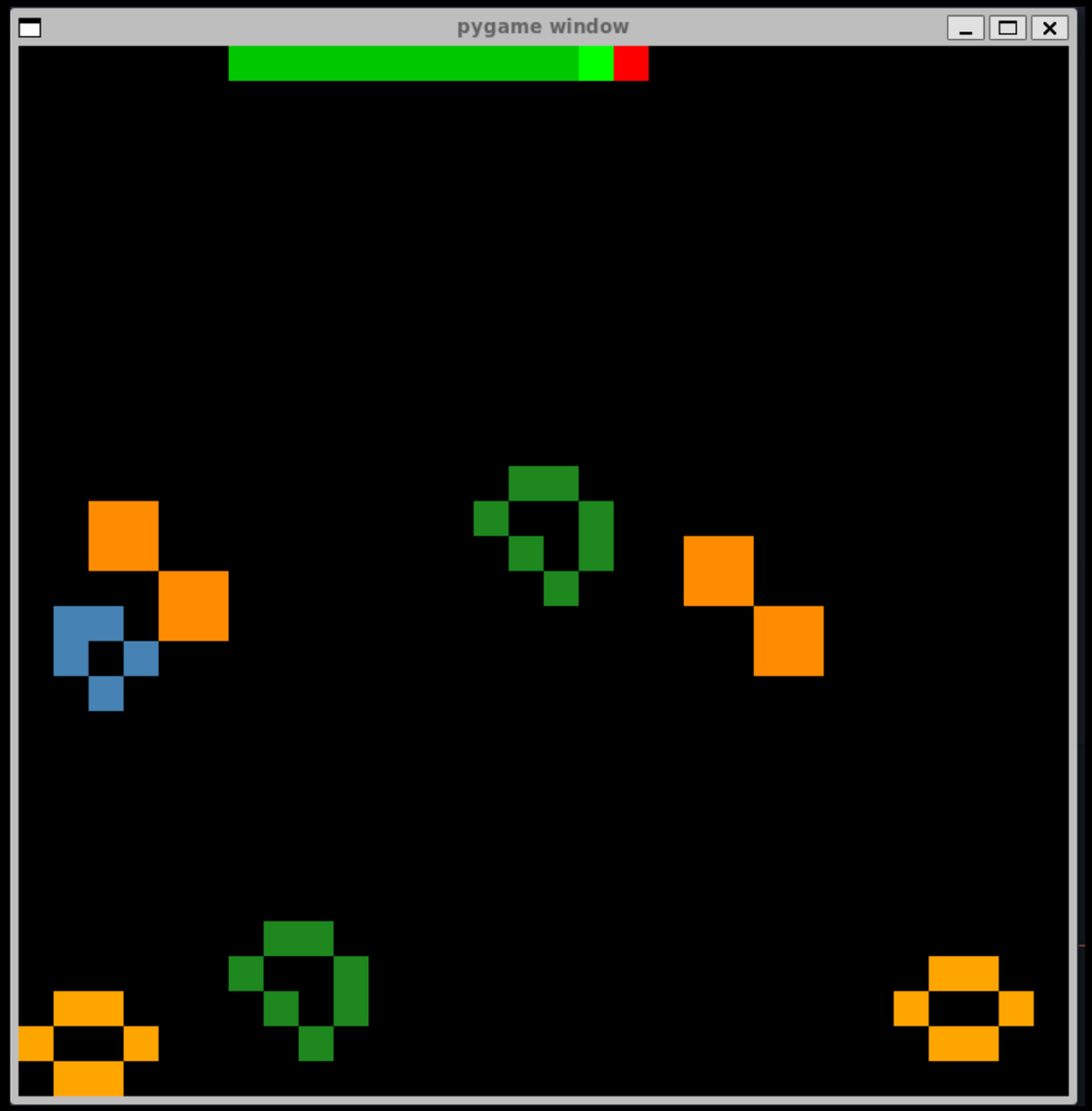
Part 3


objects
Obstacle
+max_obstacles: int
+spawn_chance: float
+despawn_chance: float
+objects: List[{cells, color, static}]
+positions: Set[(int,int)]
+reset()
+update(snake_body, food_pos)
+check_collision(pos)
+get_colored_cells()
-_neighbors8(cell)
-_evolve_all()
GridObject
+w: int
+h: int
+reset()
uses (spawn)
Snake
+alive: bool
+grow_pending: bool
+body: deque[(int,int)]
+direction: (int,int)
+reset()
+change_direction(action)
+move()
+grow()
+check_collision(pos)
Food
+value: int
+position: (int,int)
+color: (int,int,int)
+reset()
+respawn(snake_body, forbidden_positions)
EvolutionPolicy
+evolve(cells, neighbors_fn, in_bounds): Set[(int,int)]
ObstacleShape
+static: bool
+color: (int,int,int)
+shape_cells(x, y): List[(int,int)]
ConwayEvolutionPolicy
NoEvolutionPolicy
RedFood
GoldFood
BlinkerObstacle
ToadObstacle
BeaconObstacle
GliderObstacle
LShapeObstacle
CubeObstacle
BeehiveObstacle
LoafObstacle

Part 3


main_snake
py
app
main_snake
py
GameRunner
+render: bool
+obstacles: bool
+play(agent_cls, episodes, evolution_mode)
tools
SnakeEnv
+grid_size: int
+cell_size: int
+render_mode
+max_steps: int
+reset()
+step(action)
+close()
-_get_obs()
Renderer
+draw_grid()
+draw_snake()
+draw_food()
+draw_obstacles()
+flip()
policies
RewardPolicy
+compute(ate_food, died, steps, max_steps, score)
BaseMap
+grid_size: int
+create_obstacles(spawn_policy, evolution_policy)
SpawnPolicy
+max_cells: int
+spawn_chance: float
+despawn_chance: float
+with_rates(...)
ClassicRewardPolicy
+reward_food: float
+reward_death: float
+step_penalty: float
+compute(...)
ObstacleMap
+create_obstacles(...)
EmptyMap
+create_obstacles(...)

Part 3



Part 3

Obstacle Evolution Rule:

1. 對於每個活著的細胞，若身邊活著的細胞數量小於 2 個，在下個階段則會死亡
2. 對於每個活著的細胞，若身邊活著的細胞數量為 2 或 3，下個階段則會繼續活著
3. 對於每個活著的細胞，若身邊活著的細胞數量大於 3 ，下個階段則會繼續死亡
4. 對於每個死亡細胞，若身邊活著的細胞數量恰等於 3 ，下個階段則會復活