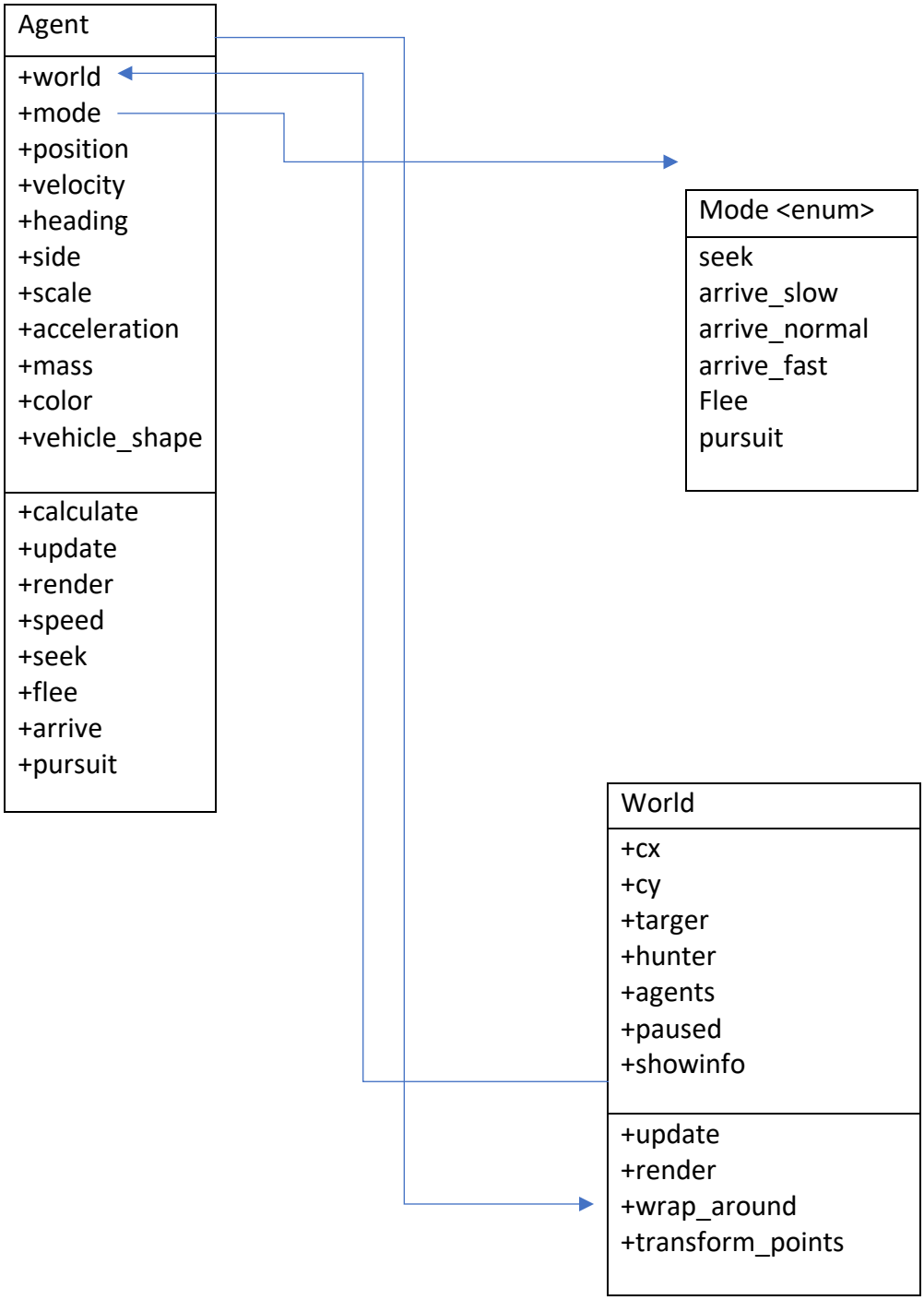
Agent
+world
+mode
+position
+velocity
+heading
+side
+scale
+acceleration
+mass
+color
+vehicle_shape
+calculate
+update
+render
+speed
+seek
+flee
+arrive
+pursuit
Mode <enum>
seek
arrive_slow
arrive_normal
arrive_fast
Flee
pursuit
World
+cx
+cy
+targer
+hunter
+agents
+paused
+showinfo
+update
+render
+wrap_around
+transform_points

Add additional agents (or multiples) to the world by pressing a key. Number of ships added at a time can be monitored.

```
def on_key_press(symbol, modifiers):
    if symbol == KEY.P:
        world.paused = not world.paused
    elif symbol == KEY.ENTER:
        for i in range(1):
            world.agents.append(Agent(world))
    elif symbol in AGENT_MODES:
        for agent in world.agents:
            agent.mode = AGENT_MODES[symbol]
```

Complete the `flee()` behaviour
Add a “panic distance” to the `flee()` code so that it only “kicks in” when the agent is close to the flee location.

```
def flee(self, hunter_pos):
        ''' move away from hunter position '''

        if(Vector2D.distance(self.pos, hunter_pos) < 100):
            desired_vel = (self.pos - hunter_pos).normalise() * self.max_speed
            return (desired_vel - self.vel)
        elif(Vector2D.distance(self.pos, hunter_pos) > 300):
            return self.seek(hunter_pos) #TODO: Replace with wander function call
        else:
            return Vector2D()
```

Add additional arrive deceleration speeds.

```
    DECELERATION_SPEEDS = {
        'slow': 0.9,
        'normal': 0.5,
        'fast': 0.1,
    }
```

Change the physical properties of the agents.

```
self.mass = 10
        # limits?
        self.max_speed = 2000.0
```

Extension

Pursuit mode is added.

```
  def pursuit(self, evader):
this behaviour predicts where an agent will be in time T and seeks
        towards that point to intercept it.
  self.toEvader = evader.pos - self.pos
      self.relHeading = Vector2D.dot(evader.heading, self.heading)

      if(Vector2D.dot(self.toEvader, self.heading) > 0 and self.relHeading < -0.95):
          return self.seek(evader.pos)

      lookAheadTime = Vector2D.length(self.toEvader) / (self.max_speed + evader.speed())
      return self.seek(evader.pos + evader.vel * lookAheadTime)
```