1 class `GameState`

The class that handles recording the state of the game, including information about all cars, the ball, and the boosts. Nothing in the `game_info` object should be modified after initiation of `GameState`.

1.1 `self.is_kickoff_pause`

Boolean. Is True while the players are free to move during kickoff, but the clock is not yet running.

1.2 `self.kickoff_position`

String. Defaults to `''Other''`, and is set and interpreted by the kickoff code. Stores which of the five standard kickoff positions the bot is in for a kickoff. Will be `''Other''` when not in a kickoff.

1.3 `self.my_team`

0 or 1. Is 0 if our bot is on the blue team, is 1 if our bot is on the orange team.

1.4 `self.team_sign`

+1 or -1. Is +1 if our bot is on the blue team, is -1 if our bot is on the orange team. This is used to mirror coordinates so that most of the code doesn't depend on which team we're on.

1.5 `self.ball`

`BallState`. Stores all information about the ball.

1.6 `self.me`

`CarState`. Stores all the information about our bot's car.

1.7 `self.my_index`

Integer. The index of our bot in the framework's list of game cars.

1.8 `self.teammates`

List. A list of `CarState`s for our teammates.

1.9 `self.opponents`

List. A list of `CarState`s for our opponents.

1.10 `self.big_boosts`

List. A list of `Boostpad`s specifically for the big boosts. This probably won't get much use in our code, and may be removed later.

1.11 `self.boosts`

List. A list of all `Boostpad`s on the field, with index matching the framework's index.

1.12 `self.game_time`

Float. The time that has passed since the beginning of the game.

1.13 `self.utils_game`

`Game`. A `Game` object from RLU. Mainly used to get the time step and initialize the RLU mechnanics and make them work. I never interface with it directly for other purposes.

1.14 `self.dt`

Float. The time elapsed since the last tick. Used for anything resembling a derivative.

1.15 `self.opponent_distance`

Float. The minimum distance from an opponent to the ball in uu.

2 class `Orientation`

The object that handles all orientation related information for game objects. Can be used as yaw-pitch-roll or a rotation matrix (eventually add quaternion support as well). WARNING: Rocket League uses the order Yaw-Pitch-Roll for the orientation of objects, but much of the code here and in the framework uses Pitch-Yaw-Roll instead.

2.1 `self.pitch`

Float between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. Rotation about an object's y-axis.

2.2 `self.yaw`

Float between $-\pi$ and π. Rotation about an object's z-axis.

2.3 `self.roll`

Float between $-\pi$ and π. Rotation about an object's x-axis.

2.4 `self.matrix`

List. A list containing the vectors `front`, `left`, and `up`. Forms a 3x3 orthogonal matrix (rotation matrix).

2.5 `self.front`

`Vec3` unit vector. The vector in the front-facing direction of the object. First column of the orientation matrix.

2.6 `self.left`

`Vec3` unit vector. The vector in the left-facing direction of the object. Second column of the orientation matrix.

2.7 `self.up`

`Vec3` unit vector. The vector in the up-facing direction of the object. Third and final column of the orientation matrix.

3 `Ball(packet, team_sign, state = None)`

`packet`: `GameTickPacket`. The packet containing all gamestate information.
`team_sign`: +1 or -1. Is +1 if we are on the Blue team, -1 if we are on the Orange team. This is used to reflect quantities so that the rest of the code is team-independent.
`state`: A framework ball prediction slice object.
`Ball` returns a `BallState` object with the parameters stored in `state` if specified, otherwise in `packet`.

4 class `BallState`

4.1 `self.pos`

`Vec3`. The (x, y, z) position of the ball.

4.2 `self.rot`

`Orientation`. The `Orientation` object corresponding to the ball's current 3d orientation.

4.3 `self.vel`

`Vec3`. The (v_x, v_y, v_z) velocity of the ball.

4.4 `self.omega`

`Vec3`. The $(\omega_x, \omega_y, \omega_z)$ angular velocity of the ball.

4.5 `self.last_touch`

Is currently broken in framework. Also looks broken in the `GameState` code, or at least is not in terms of CowBot data types.

4.6 `self.hit_location`

Is currently broken in framework. Also looks broken in the `GameState` code, or at least is not in terms of CowBot data types.

4.7 `self.copy_state`

4.7.1 `pos`

`Vec3`. The (x, y, z) position of the ball.

4.7.2 `rot`

`Orientation`. The `Orientation` object corresponding to the ball's current 3d orientation.

4.7.3 `vel`

`Vec3`. The (v_x, v_y, v_z) velocity of the ball.

4.7.4 `omega`

`Vec3`. The $(\omega_x, \omega_y, \omega_z)$ angular velocity of the ball.

5 Car

6 class CarState

6.1 `self.pos`

`Vec3`. The (x, y, z) position of the car.

6.2 `self.rot`

`Orientation`. The `Orientation` object corresponding to the car's current 3d orientation.

6.3 `self.vel`

`Vec3`. The (v_x, v_y, v_z) velocity of the car.

6.4 `self.omega`

`Vec3`. The $(\omega_x, \omega_y, \omega_z)$ angular velocity of the car.

6.5 `self.demo`

Boolean. True if the car is currently off the field as a result of being demoed.

6.6 `self.wheel_contact`

Boolean. True if the car has wheel contact. Can be with the ground, walls, ceiling, ball, or another car.

6.7 `self.supersonic`

Boolean. True if the car is supersonic.

6.8 `self.jumped`

Boolean. True if the car has jumped and has not yet regained wheel contact.

6.9 `self.double_jumped`

Boolean. True if the car has double jumped, including dodges, and has not yet regained wheel contact.

6.10 `self.boost`

Float. The amount of boost that the car has.

6.11 `self.index`

Integer. The index of the car in the framework's game_cars list.

6.12 `self.jumped_last_frame`

Boolean. True if the car input jump on the previous tick. Only for our car, since we aren't allowed to look at the inputs of other cars.

6.13 `self.copy_state`

6.13.1 `pos`

`Vec3`. The (x, y, z) position of the car.

6.13.2 `rot`

`Orientation`. The `Orientation` object corresponding to the car's current 3d orientation.

6.13.3 `vel`

`Vec3`. The (v_x, v_y, v_z) velocity of the car.

6.13.4 `omega`

`Vec3`. The $(\omega_x, \omega_y, \omega_z)$ angular velocity of the car.

7 class `Boostpad`

7.1 `self.index`

Integer. The index of the boost pad in the list of boost pads.

7.2 `self.pos`

`Vec3`. The (x, y, z) position of the boost pad.

7.3 `self.is_active`

Boolean. True if the boost is currently available to be picked up.

7.4 `self.timer`

Float. The time until `is_active` becomes True again.