Projection

Module `projection`

Functions

Function `draw_scene`

```
def draw_scene(
    state: projection.State
) -> numpy.ndarray
```

Read the dataset and draw the scene

Args

`state` State object

Returns

np.ndarray Image of the scene

Function `draw_scene_newDataset`

```
def draw_scene_newDataset(
    state: projection.State
) -> numpy.ndarray
```

Read the dataset and draw the scene

Args

`state` State object

Returns

np.ndarray Image of the scene

Function `draw_scene_oldDataset`

```
def draw_scene_oldDataset(
    state: projection.State
) -> numpy.ndarray
```

Read the dataset and draw the scene

Args

`state` State object

Returns

np.ndarray Image of the scene

Function `main`

```
def main()
```

Function `project_bbox3D_img`

```
def project_bbox3D_img(
    bbox: bbox.BoundingBox,
    camera_pose: numpy.ndarray,
    camera_matrix: numpy.ndarray,
    img: numpy.ndarray
) -> numpy.ndarray
```

Project a 3D bounding box in the image plane

Args

`bbox` BoundingBox object
`camera_pose` Camera pose
`camera_matrix` Camera matrix
`img` Image to draw on

Returns

np.ndarray Image with the bounding box projected

Classes

Class `State`

```
class State(
    datasets_folder: str = '/Users/caillotantoine/Datasets',
    dataset_name: str = 'CARLA_Dataset_A',
    frame: int = 180
)
```

Class variables

Variable `dataset_name` Type: `str`

Name of the dataset

Variable `datasets_folder` Type: `str`

Path to the dataset folder

Variable `frame` Type: `int`

Frame to observe

Variable `mutex` Type: `<built-in function allocate_lock>`

Mutex to read and write the State object

Methods

Method `read_vars`

```
def read_vars(
    self
) -> Tuple[str, str, int]
```

Read the State object

Returns

(str, str, int): (datasets_folder, dataset_name, frame)
