Rectangle_2D_Description

May 25, 2024

[ ]:
```python
from Point_2D import *


class Rectangle:
    """2D Rectangle Class"""

    def __init__(self, corner_lb: Point, width: float, height: float):
        self.corner = corner_lb
        self.width = width
        self.height = height
        self.corners = {
            "corner_lb": self.corner,
            "corner_rt": Point(self.corner.x + self.width, self.corner.y + self.
 ↪height),
            "corner_rb": Point(self.corner.x + self.width, self.corner.y),
            "corner_lt": Point(self.corner.x, self.corner.y + self.height),
        }

    def __repr__(self) -> str:
        return f"Rectangle({self.corner}, {self.width}, {self.height})"

    def __str__(self) -> str:
        return f"[Corner: {self.corner}, Width: {self.width}, Height: {self.
 ↪height}]"

    def __mul__(self, lamda: float) -> "Rectangle":
        """Scales the rectangle by the given factor `lamda`."""
        return Rectangle(self.corner, self.width * lamda, self.height * lamda)

    def __eq__(self, other: "Rectangle") -> bool:
        """Checks for strict equality between two rectangles."""
        return (
            self.corner == other.corner
            and self.width == other.width
            and self.height == other.height
        )
```

```
    def is_same(self, other: "Rectangle") -> bool:
        """Checks for geometric equality between two rectangles."""
        return self.width == other.width and self.height == other.height

    def is_similar(self, other: "Rectangle") -> bool:
        """Checks for geometric similarity between two rectangles."""
        return self.width / self.height == other.width / other.height

    def grow(self, dw: float, dh: float) -> "Rectangle":
        """Returns a new rectangle grown by `dw` in width and `dh` in height."""
        return Rectangle(self.corner, self.width + dw, self.height + dh)

    def move(self, dx: float, dy: float) -> "Rectangle":
        """Returns a new rectangle moved by `dx` along the x-axis and `dy`␣
↪along the y-axis."""
        new_corner = Point(self.corner.x + dx, self.corner.y + dy)
        return Rectangle(new_corner, self.width, self.height)

    def area(self) -> float:
        """Returns the area of the rectangle."""
        return self.width * self.height

    def perimeter(self) -> float:
        """Returns the perimeter of the rectangle."""
        return 2 * (self.width + self.height)

    def flip(self) -> "Rectangle":
        """Returns a new rectangle with the width and height swapped."""
        return Rectangle(self.corner, self.height, self.width)

    def contain(self, other: "Point") -> bool:
        """Determines whether the rectangle contains the given point."""
        return (
            self.corner.x <= other.x <= self.corner.x + self.width
            and self.corner.y <= other.y <= self.corner.y + self.height
        )

    def collide(self, other: "Rectangle") -> bool:
        """Determines whether two rectangles collide with each other."""
        flag = False
        # Check whether there exists a corner that is contained within the␣
↪other rectangle.
        for corner in self.corners.values():
            flag = flag or other.contain(corner)
        for corner in other.corners.values():
            flag = flag or self.contain(corner)
        # Check whether the diagonals intersect.
```

```
        flag = flag or intersect(
            self.corner,
            self.corners["corner_rt"],
            other.corner,
            other.corners["corner_rt"],
        )
        return flag


a1 = Point(1, 2)
a2 = Point(-2, 3)
R1 = Rectangle(a1, 10, 20)
R2 = Rectangle(a2, 20, 2)

print(R1.collide(R2))
```

```
True
```

[ ]:
```
help(Rectangle)
```

```
Help on class Rectangle in module __main__:

class Rectangle(builtins.object)
 |  Rectangle(corner_lb: Point_2D.Point, width: float, height: float)
 |
 |  2D Rectangle Class
 |
 |  Methods defined here:
 |
 |  __eq__(self, other: 'Rectangle') -> bool
 |      Checks for strict equality between two rectangles.
 |
 |  __init__(self, corner_lb: Point_2D.Point, width: float, height: float)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __mul__(self, lamda: float) -> 'Rectangle'
 |      Scales the rectangle by the given factor `lamda`.
 |
 |  __repr__(self) -> str
 |      Return repr(self).
 |
 |  __str__(self) -> str
 |      Return str(self).
 |
 |  area(self) -> float
 |      Returns the area of the rectangle.
 |
 |  collide(self, other: 'Rectangle') -> bool
```

```
 |      Determines whether two rectangles collide with each other.
 |
 |  contain(self, other: 'Point') -> bool
 |      Determines whether the rectangle contains the given point.
 |
 |  flip(self) -> 'Rectangle'
 |      Returns a new rectangle with the width and height swapped.
 |
 |  grow(self, dw: float, dh: float) -> 'Rectangle'
 |      Returns a new rectangle grown by `dw` in width and `dh` in height.
 |
 |  is_same(self, other: 'Rectangle') -> bool
 |      Checks for geometric equality between two rectangles.
 |
 |  is_similar(self, other: 'Rectangle') -> bool
 |      Checks for geometric similarity between two rectangles.
 |
 |  move(self, dx: float, dy: float) -> 'Rectangle'
 |      Returns a new rectangle moved by `dx` along the x-axis and `dy` along
the y-axis.
 |
 |  perimeter(self) -> float
 |      Returns the perimeter of the rectangle.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```