Bus.cs

This class represents a bus and the power transmission quantities associated with one.
It is derived from the class `Sprite` which defines loading content and positioning.

Class Fields

Name	Type	Explanation
powerbar	`Texture2D`	The image of the status bar above the bus.
blades	`Texture2D`	In case the bus is a wind turbine, this texture is the image of the blades that will be rotated.
bladesPosition	`Vector2`	The position of the wind turbine blades.
bladesAngle	`float`	The rotation angle of the wind turbine blades.
sizeFrame	`Rectangle`	This is the rectangle that selects the frame part of the `powerbar` spritesheet.
sizeBar	`Rectangle`	This is the rectangle that selects the bar part of the `powerbar` spritesheet.
barPos	`Vector2`	A Vector that describes the position of the status bar.
id	`int`	The ID of the bus.
p	`double`	The power of the bus.
pMax	`double`	The maximal power the bus can reach.
pMin	`double`	The minimal power the bus can reach. It the bus is a generator then `pMin` is the maximal generation the bus can produce.
mouseRectangle	`Rectangle`	A rectangle enclosing the graphics. It is used to check for mouseclicks.
u	`double`	The voltage magnitude of the bus. `Not in use`
theta	`double`	The voltage angle of the bus.

Name	Type	Explanation
type	`string`	The type of the bus. It can be either a generator “G”, a hydro power plant “H”, a photovoltaic power plant “P”, a wind turbine “W”, a consumer “C” or a connection line “S”.
pForBattery	`double`	It equals p as long as there is no battery. If there is a battery it is the power the bus would have without the battery.

Class constructor

```
public Bus(Vector2 position, int Id, double P, double PMax, double
PMin, double U, double Theta, string Type, double PForBattery,
GraphicsDevice gDevice) : base(position, gDevice)
        {
            this.PMax = PMax;
            this.PMin = PMin;
            this.Type = Type;
            this.Id = Id;
            this.P = P;
            this.U = U;
            this.Theta = Theta;
            this.PForBattery = PForBattery;
        }
```

Input Arguments

- `position`: The position of the bus.
- `ID`: The ID of the bus.
- `P`: The initial power of the bus.
- `PMax`: The maximum power of the bus.
- `PMin`: The minimum power of the bus. If the bus is a generator this is the maximum generation (as it is negative).
- `U`: The voltage magnitude. Currently not in use.
- `Theta`: The intial angle theta.
- `Type`: The type of the bus. See field description for details.
- `gDevice`: An instance of the graphics device. Used for scaling the image in the base class.

Description

The constructor is very basic and just assigns all input values to their respective fields.
PForBattery is assigned to P as it is a reference point when Bus power is changed by a battery charging/discharging.

Properties

- Id
- P
- PMax
- PMin
- MouseRectangle
- U
- Theta
- Type
- PForBattery

Id

```
public int Id
```

Set

Sets its value to id.

Get

Returns the value of id.

P

```
public double P
```

Set

Sets its value to PMax if the input value is bigger than PMax. Sets its value to PMin if the input value is smaller than PMin. Sets its value to p in all other cases.

Get

Returns the value of p.

PMax

```
public double PMax
```

Set

Sets its value to pMax.

Get

Returns the value of pMax.

PMin

```
public double PMin
```

Set

Sets its value to pMin.

Get

Returns the value of pMin.

MouseRectangle

```
public Rectangle MouseRectangle
```

Set

Sets its value to mouseRectangle.

Get

Returns the value of mouseRectangle.

U

```
public double U
```

Set

Sets its value to u.

Get

Returns the value of u.

Theta

```
public double Theta
```

Set

Sets its value to theta. If theta is bigger that 360 than it sets its value theta % 360.

Get

Returns the value of theta.

Type

```
public string Type
```

Set

Sets its value to type.

Get

Returns the value of type.

PForBattery

```
public double PForBattery
```

Set

Sets its value to pForBattery.

Get

Returns the value of pForBattery.

Methods

```
public void loadBar(ContentManager cManager)
public void loadBlades(ContentManager cManager)
public void setMouseRectangle()
public String toString()
public void Draw(SpriteBatch spriteBatch, SpriteFont theText)
```

loadBar

```
public void loadBar(ContentManager cManager)
```

Input Arguments

- `cManager`:

Description

This method loads the graphics required for the status bar above the bus.
It also defines the rectangles which select the appropriate part of the spritesheet loaded.
The bar is positioned 10 pixel above the bus image.

---

loadBlades

```
public void loadBlades(ContentManager cManager)
```

Input Arguments

- `cManager`: An instance of the current content manager.

Description

This method can be used to load the image of the blades in case the bus is a wind generator.
This is necessary as the draw function draws the blades image, otherwise the texture would be `null`.

---

setMouseRectangle

```
public void setMouseRectangle()
```

Input Arguments

None.

Description

This method sets the rectangle that describes the bounds of the bus image.
This is done so `PowerModel` can check if the bus has been clicked on by the player.

---

toString

```
public String toString()
```

Input Arguments

None.

Description

This method assembles a string of relevant information about the bus (namely ID, current P along with the max/min and Theta).

Returns

A string containing information about the bus.

Draw

```
public void Draw(SpriteBatch spriteBatch, SpriteFont theText)
```

Input Arguments

- `spriteBatch`: The `SpriteBatch` used to draw images.

- `theText`: The `SpriteFont` used to draw text.

Description

This method is called by `PowerModel` every frame, 60 times a second.
First it draws the Texture of the Bus in the correct position.
Then if the bus is a wind generator, it will draw the blades on top of the tower and rotate it up to 0.01f rad per 1/60th of a second, depending on how much power the bus is generating.
The status bar above the bus is colored depending on whether it is a generator or consumer.

- Generator: Green
- Consumer: Orange

`int barSize = (int)((Math.Abs((P - PMin))) / Math.Abs((PMax - PMin)) * 60d);` evaluates how "full" the bar is. It multiplies a ratio between 0 and 1 with its width (60 pixels).
If the bus is a generator this expression yields 0 when it generates at full power, therefore 60pixels `- barSize` yields the correct result.

Finally it draws the BusID underneath the bus.