Battery.cs

This class includes all the values of a battery. It also decides when the battery cannot be charged or discharged.
It inherits from Sprite.

Class Fields

Name	Type	Explanation
powerbar	`Texture2D`	The image of the status bar above the bus.
lineTexture	`Texture2D`	The 1x1 pixel Texture that is stretched to represent a line connecting to the bus.
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.
chargeState	`PowerModel.BatteryChargeState`	The charge state of the battery. It is either `Idle`, `Charging` or `Discharging`.
chargeStartTime	`float`	Time since the `chargeState` changed to its momentary state.
startCharge	`double`	`E` at the time the `chargeState` changed.
chargeRate	`double`	The charge rate of the battery defined in the level.
power	`double`	The power by which the bus connected to the battery is going to change if the battery is charged or discharged.
busPos	`Vector2`	The position of the bus connected to the battery.
allowsChange	`bool`	It is `true` if manual change of the battery is allowed. It is `false` if manual change is not allowed.

Name	Type	Explanation
e	`double`	The energy which the battery has stored.
eMax	`double`	The maximal energy which the battery can store.
busID	`int`	The ID of the bus to which the battery is connected.
mouseRectangle	`Rectangle`	A rectangle enclosing the graphics. It is used to check for mouseclicks.

Class constructor

```
public Battery(int busId, double power, double eMax, double initialCharge,
double chargeRate, Vector2 position, GraphicsDevice gDevice,
bool allowsChange) : base(position + new Vector2(70, 70), gDevice)
        {
            this.BusID = busId;
            this.EMax = eMax;
            this.chargeState = PowerModel.BatteryChargeState.Idle;
            this.chargeStartTime = 0f;
            this.E = initialCharge;
            this.chargeRate = chargeRate;
            this.busPos = position;
            this.power = power;
            this.allowsChange = allowsChange;
        }
```

Input Arguments

- `busID`: The ID of the corresponding bus.
- `power`: The amount of power the battery can consume or produce.
- `eMax`: The maximum energy capacity.
- `initialCharge`: The initial charge of the battery when the level starts.
- `chargeRate`: The charging rate of the battery. 1 corresponds to 1 Energy per Hour.
- `position`: The position of the battery. This gets passed to the base class but moved by (70,70) so it doesn't overlap with the bus image.
- `gDevice`: The current graphics device.
- `allowsChange`: A boolean describing if the battery is interactive or not.

Description

This constructor sets the provided parameters to the corresponding elements in the battery.

Properties

- CurrentCharge
- Power
- ChargeCapacity
- AllowsChange
- ChargeState
- BusID
- E
- EMax
- MouseRectangle

CurrentCharge

```
public double CurrentCharge
```

Set

Empty.

Get

Returns the value of startCharge.

Power

```
public double Power
```

Set

Empty.

Get

Returns the value of power.

ChargeCapacity

```
public double ChargeCapacity
```

Set

Empty.

Get

Returns the value of chargeCapacity.

AllowsChange

```
public bool AllowsChange
```

Set

Sets its value to allowsChange.

Get

Returns the value of allowsChange.

ChargeState

```
public PowerModel.BatteryChargeState ChargeState
```

Get

Returns the value of chargeState.

BusID

```
public int BusID
```

Set

Sets its value to busID.

Get

Returns the value of busID.

E

```
public double E
```

Set

Sets its value to 0 if the input value is smaller or equal to 0. Sets its value to eMax if the input value is bigger or equal to eMax. Sets its value to e in all other cases.

Get

Returns the value of e.

EMax

```
public double EMax
```

Set

Sets its value to 0 if the input value is smaller or equal to 0. Sets its value to eMax in all other cases.

Get

Returns the value of eMax.

MouseRectangle

```
public Rectangle MouseRectangle
```

Set

Sets its value to mouseRectangle.

Get

Returns the value of mouseRectangle.

Methods

```
public void setMouseRectangle()
public void loadBar(ContentManager cManager)
public void Update(float elapsedTime)
public void charge(float elapsedTime)
public void discharge(float elapsedTime)
public void changeChargeState(PowerModel.BatteryChargeState newState, float startTime)
public void loadLineTexture(Texture2D texture)
public string toString()
private void drawLineToBus(SpriteBatch spriteBatch)
new public void Draw(SpriteBatch spriteBatch)
```

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.

---

loadBar

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

Input Arguments

- `cManager`: An instance of the games current `ContentManager`.

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.

Update

```
public void Update(float elapsedTime)
```

Input Arguments

- `elapsedTime`: The elapsed time in minutes from the power model.

Description

The update function of a battery is called 60 times a second by `PowerModel`.
Depending on `chargeState` it will either charge or discharge the battery.

---

charge

```
public void charge(float elapsedTime)
```

Input Arguments

- `elapsedTime`: The elapsed time in minutes from the power model.

Description

This method handles charging the battery.
If the level sets the battery to `chargeCapacity = 1`, this will charge the battery by 1 unit of energy in an hour.

---

discharge

```
public void discharge(float elapsedTime)
```

Input Arguments

- `elapsedTime`: The elapsed time in minutes from the power model.

Description

This method handles discharging the battery.
If the level sets the battery to `chargeCapacity = 1`, this will discharge the battery by 1 unit of energy in an hour.

---

changeChargeState

```
public void changeChargeState(PowerModel.BatteryChargeState newState, float startTime)
```

Input Arguments

- `newState`: The new `ChargeState` the battery should be set to.
- `startTime`: The `PowerModel` level time when the state change was made.

Description

This method allows the user to change the battery state by hand.
If the new state is the same as the old state, nothing should happen.
Then it sets `chargeStartTime` to `startTime` as a reference point when the charging/discharging started.
After that it sets the current chargeState to the new one and finally sets `startCharge` to the current charge.

Returns

Describe what this function returns and what the return value is used for.

---

loadLineTexture

```
public void loadLineTexture(Texture2D texture)
```

Input Arguments

- `texture`: The texture to be saved

Description

This method saves the 1x1 px line texture used to drawing a line to the associated bus.
This is necessary to call because the texture should not be `null`.

---

toString

```
public String toString()
```

Input Arguments

None.

Description

This method assembles a string of relevant information about the battery (namely corresponding bus ID, current capacity along with its maximum capacity and the current charge state).

Returns

A string containing information about the battery.

---

drawLineToBus

```
private void drawLineToBus(SpriteBatch spriteBatch)
```

Input Arguments

- `spriteBatch`: The SpriteBatch used for drawing.

Description

This method draws a line to the bus where the battery is attached.
First it calculates the distance vector `edge` from the battery position to the bus position, then gets the angle from the x-Axis via

```
float angle = (float)Math.Atan2(edge.Y, edge.X);
```

and finally draws the line by stretching the 1x1 px line texture to the length of the distance vector, then rotating it
by `angle` to point it in the direction of the bus.

Returns

Describe what this function returns and what the return value is used for.

Draw

```
new public void Draw(SpriteBatch spriteBatch)
```

Input Arguments

- `spriteBatch`: The spritebatch used for drawing.

Description

This method draws all images related to the battery.
First it draws the battery image, then it decides on a color for the status bar depending on `ChargeState`:

- Charging: Green
- Discharging: Red
- Idle: Yellow

It then draws the frame of the status bar and sizes the filling according to `int barSize = (int)((E) / EMax * 60d);`. The bar is 60 pixel wide and E/EMax describes a ratio between 0 and 1.

Finally it calls `drawLineToBus(spriteBatch)` to draw a line to the connecting bus.