Snake

Generated by Doxygen 1.10.0

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 software::game Class Reference

Public Member Functions

- game (hardware::led_matrix _ledMatrix, hardware::joy_stick _joyStick)

  Construct a new game object.
- ~**game** ()

  Destroy the game object.
- void **init** ()

  Should be called shortly after the object is constructed. Initializes the ledMatrix_ object and the joyStick_ object by calling their init functions. Unsets all pixels in gameBoard_. Sets a random inital direction and a random start point for the snake. The random startpoint is in an inner square which is 70% of the game board size. Spawns 2 randomly placed foods on the board.
- bool exec ()

  Executes one frame of the game, whereby a frame and a move of the snake is the same cycle. Thus the game speed is controlled by the rate of calling exec. Should be called about every 300 ms for playable results. Returns if the game is lost.
- bool menu ()

  Not yet implemented. Should not be used. At current state just returns if the joy stick is pressed.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 game()

```
software::game::game (
            hardware::led_matrix _ledMatrix,
            hardware::joy_stick _joyStick )
```

Construct a new game object.

Parameters

_ledMatrix	Object for storing the LED information, setting colours and controlling higher graphical functionality.
_joyStick	Object for getting directional information of the connected joy stick.

3.1.2 Member Function Documentation

3.1.2.1 exec()

```
bool software::game::exec ( )
```

Executes one frame of the game, whereby a frame and a move of the snake is the same cycle. Thus the game speed is controlled by the rate of calling exec. Should be called about every 300 ms for playable results. Returns if the game is lost.

Returns

true Game can continue.

false Game is lost.

3.1.2.2 menu()

```
bool software::game::menu ( )
```

Not yet implemented. Should not be used. At current state just returns if the joy stick is pressed.

Returns

true Joy stick is pressed.

false Joy stick is not pressed.

The documentation for this class was generated from the following files:

- game.hpp
- game.cpp

3.2 hardware::joy_stick Class Reference

Public Types

- enum direction {
  noDirection , **up** , **down** , **left** ,
  right }
  Enum for the possible inputs of the joy stick.

Public Member Functions

- joy_stick (uint8_t _buttonPin, uint8_t _vrxPin, uint8_t _vryPin)
  Construct a new joy stick object.
- joy_stick (joy_stick &t)
  Copy constructor for a new joy stick object.
- ~**joy_stick** ()
  Destroy the joy stick object.
- void **init** ()
  Init function for this object. Need to be called shortly after initialisation.
- direction getDirection ()
  Get the current direction of the joy stick.
- bool isPressed ()
  Returns if the button is currently pressed.

3.2.1 Constructor & Destructor Documentation

3.2.1.1 joy_stick() [1/2]

```
hardware::joy_stick::joy_stick (
            uint8_t _buttonPin,
            uint8_t _vrxPin,
            uint8_t _vryPin )
```

Construct a new joy stick object.

Parameters

_buttonPin	Pin number of the button pin.
_vrxPin	Pin number of the vrx pin.
_vryPin	Pin number of the vry pin.

3.2.1.2 joy_stick() [2/2]

```
hardware::joy_stick::joy_stick (
            joy_stick & t )
```

Copy constructor for a new joy stick object.

Parameters

t	Object of joy_stick to copy from.

3.2.2 Member Function Documentation

3.2.2.1 getDirection()

```
hardware::joy_stick::direction hardware::joy_stick::getDirection ( )
```

Get the current direction of the joy stick.

Returns

direction Enum of all possible inputs.

3.2.2.2 isPressed()

```
bool hardware::joy_stick::isPressed ( )
```

Returns if the button is currently pressed.

Returns

true Button is pressed.

false Button is not pressed.

The documentation for this class was generated from the following files:

- joy_stick.hpp
- joy_stick.cpp

3.3 hardware::led_matrix Class Reference

Public Types

- enum colorPixel { **obstacle** = CRGB::Red , **tail** = CRGB::Yellow , **food** = CRGB::Green , **head** = CRGB::Blue }

  Enumeration of the default colors for every game pixel.

Public Member Functions

- **led_matrix** ()

  Construct a new led matrix object.
- led_matrix (led_matrix &t)

  Copy construct for a new led matrix object.
- ∼**led_matrix** ()

  Destroy the led matrix object.
- void **init** ()

  Init function for this object. Need to be called shortly after initialisation.
- void setPixel (uint16_t x, uint16_t y, colorPixel color)

  Set the color from the default colors of a coordinate.
- void setPixel (uint16_t x, uint16_t y, uint32_t color)

  Set the color of a coordinate.
- void unsetPixel (uint16_t x, uint16_t y)

  Unsets the pixel by setting the color value to 0 making the LED black.
- void printText (const char ∗text, int16_t x, int16_t y, CRGB::HTMLColorCode color, bool wrap)

  Print text on the led matrix. Uses FastLED Neomatrix library. Text could wrap and scroll.
- void setBrightness (uint8_t brightn)

  Set the brightness of the led matrix.
- void **outputMatrix** ()

  Outputs the content of the leds_ array to the physical led matrix.
- void **clearMatrix** ()

  Clears the content in leds_ matrix. If this should take effect on the physical matrix, outputMatrix() should be called aferwards.

Protected Attributes

- uint8_t **brightness_** {40}

  Brightness of the leds on the matrix. Should not be too low, to avoid weird side effects. Too high values could also lead to weird side effects or just not working matrix, due to not enough power delivery.
- bool **initailized_** {false}

  Is set true, when init() is called.
- CRGB **leds_** [NUM_LEDS]

  Matrix of CRGB objects for setting the color values of the LEDs.
- FastLED_NeoMatrix * **matrix_**

  Object of the FastLED Neomatrix library for controlling LEDs. This is used for higher graphical functionality like scrolling text for example.

3.3.1 Constructor & Destructor Documentation

3.3.1.1 led_matrix()

```
hardware::led_matrix::led_matrix (
            led_matrix & t )
```

Copy construct for a new led matrix object.

Parameters

t	Object to copy from.

3.3.2 Member Function Documentation

3.3.2.1 printText()

```
void hardware::led_matrix::printText (
            const char * text,
            int16_t x,
            int16_t y,
            CRGB::HTMLColorCode color,
            bool wrap )
```

Print text on the led matrix. Uses FastLED Neomatrix library. Text could wrap and scroll.

Parameters

text	Character array of the text to print.
x	TODO where is the starting point?
y	TODO
color	TODO what does this option?
wrap	If true, text wraps around edges.

3.3.2.2 setBrightness()

```
void hardware::led_matrix::setBrightness (
            uint8_t brightn )
```

Set the brightness of the led matrix.

Parameters

brightn	Brightness to set.

3.3.2.3 setPixel() [1/2]

```
void hardware::led_matrix::setPixel (
            uint16_t x,
            uint16_t y,
            colorPixel color )
```

Set the color from the default colors of a coordinate.

Parameters

x	X value of the coordinate to set.
y	Y value of the coordinate to set.
color	Default color enum.

3.3.2.4 setPixel() [2/2]

```
void hardware::led_matrix::setPixel (
            uint16_t x,
            uint16_t y,
            uint32_t color )
```

Set the color of a coordinate.

Parameters

x	X value of the coordinate to set.
y	Y value of the coordinate to set.
color	Color value.

3.3.2.5 unsetPixel()

```
void hardware::led_matrix::unsetPixel (
            uint16_t x,
            uint16_t y )
```

Unsets the pixel by setting the color value to 0 making the LED black.

Parameters

x	X value of the coordinate to set.
y	Y value of the coordinate to set.

The documentation for this class was generated from the following files:

- led_matrix.hpp
- led_matrix.cpp

3.4 hardware::pixelCoordinate Struct Reference

Public Member Functions

- bool increaseX ()

  Increases the x_ value by 1, but not higher than the MATRIX_HEIGHT.
- bool increaseY ()

  Increases the y_ value by 1, but not higher than the MATRIX_WIDTH.
- bool decreaseX ()

  Decreases the x_ value by 1, but not less than 0.
- bool decreaseY ()

  Decreases the y_ value by 1, but not less than 0.

Public Attributes

- uint8_t **x_**
- uint8_t **y_**

3.4.1 Member Function Documentation

3.4.1.1 decreaseX()

```
bool hardware::pixelCoordinate::decreaseX ( )
```

Decreases the x_ value by 1, but not less than 0.

Returns

true New value is not out of matrix dimension.

false New value is out of matrix dimension.

3.4.1.2 decreaseY()

```
bool hardware::pixelCoordinate::decreaseY ( )
```

Decreases the y_ value by 1, but not less than 0.

Returns

true New value is not out of matrix dimension.
false New value is out of matrix dimension.

3.4.1.3 increaseX()

```
bool hardware::pixelCoordinate::increaseX ( )
```

Increases the x_ value by 1, but not higher than the MATRIX_HEIGHT.

Returns

true New value is not out of matrix dimension MATRIX_HEIGHT.
false New value is out of matrix dimension MATRIX_HEIGHT.

3.4.1.4 increaseY()

```
bool hardware::pixelCoordinate::increaseY ( )
```

Increases the y_ value by 1, but not higher than the MATRIX_WIDTH.

Returns

true New value is not out of matrix dimension MATRIX_WIDTH.
false New value is out of matrix dimension MATRIX_WIDTH.

The documentation for this struct was generated from the following files:

- led_matrix.hpp
- led_matrix.cpp

3.5 software::snake Class Reference

Public Member Functions

- **snake** ()

  Construct a new snake object.
- ∼**snake** ()

  Destroy the snake object.
- void init (hardware::pixelCoordinate coord, hardware::joy_stick::direction dir)

  Init function for this object. Need to be called shortly after initialisation. The start coordinate and start direction is set by this method.
- void move (hardware::pixelCoordinate coord)

  Move the snake on coordinate. Add the new coordinate and deletes the tail coordiante of the snake.
- void grow (hardware::pixelCoordinate coord)

  Grows the snake to the coordinate. Add the new coordinate, but does not delete the tail coordinate.
- hardware::pixelCoordinate getHeadCoord ()

  Get the coordinate of the snake head.
- hardware::pixelCoordinate getTailCoord ()

  Get the coordinate of the snake tail.
- hardware::joy_stick::direction getDirection ()

  Get the direction of the snake.
- void setDirection (hardware::joy_stick::direction dir)

  Set the direction object of the snake. Can not be no direction, must be up, right, lef or down.
- uint8_t getSnakeLength ()

  Get the length of the snake, the number of coordinates in the deque.

3.5.1 Member Function Documentation

3.5.1.1 getDirection()

```
hardware::joy_stick::direction software::snake::getDirection ( )
```

Get the direction of the snake.

Returns

hardware::joy_stick::direction Direction of the snake.

3.5.1.2 getHeadCoord()

```
hardware::pixelCoordinate software::snake::getHeadCoord ( )
```

Get the coordinate of the snake head.

Returns

hardware::pixelCoordinate Coordinate of the head.

3.5.1.3 getSnakeLength()

```
uint8_t software::snake::getSnakeLength ( )
```

Get the length of the snake, the number of coordinates in the deque.

Returns

uint8_t Length of the snake.

3.5.1.4 getTailCoord()

```
hardware::pixelCoordinate software::snake::getTailCoord ( )
```

Get the coordinate of the snake tail.

Returns

hardware::pixelCoordinate Coordinate of the tail.

3.5.1.5 grow()

```
void software::snake::grow (
            hardware::pixelCoordinate coord )
```

Grows the snake to the coordinate. Add the new coordinate, but does not delete the tail coordinate.

Parameters

coord	New coordinate to add.

3.5.1.6 init()

```
void software::snake::init (
            hardware::pixelCoordinate coord,
            hardware::joy_stick::direction dir )
```

Init function for this object. Need to be called shortly after initialisation. The start coordinate and start direction is set by this method.

Parameters

coord	Starting coordinate.
dir	Starting direction.

3.5.1.7 move()

```
void software::snake::move (
            hardware::pixelCoordinate coord )
```

Move the snake on coordinate. Add the new coordinate and deletes the tail coordiante of the snake.

Parameters

coord	New coordinate to add.

3.5.1.8 setDirection()

```
void software::snake::setDirection (
            hardware::joy_stick::direction dir )
```

Set the direction object of the snake. Can not be no direction, must be up, right, lef or down.

Parameters

dir	New direction to set.

The documentation for this class was generated from the following files:

- snake.hpp
- snake.cpp

Chapter 4

File Documentation

4.1 game.hpp

```
00001 #ifndef __game_header_included__
00002 #define __game_header_included__
00003
00004 #include <string>
00005
00006 #include "snake.hpp"
00007 #include "led_matrix.hpp"
00008 #include "joy_stick.hpp"
00009
00010 namespace software {
00011
00012 class game {
00013   private:
00019     enum gameMode {
00020       easy,
00021       normal,
00022       hard
00023     };
00028     enum pixelType {
00029       unset,
00030       snakeHead,
00031       snakeTail,
00032       food,
00033       obstacle
00034     };
00035
00036     //std::array<std::array<pixelType, MATRIX_WIDTH>, MATRIX_HEIGHT> gameBoard_;
00042     pixelType gameBoard_[MATRIX_WIDTH][MATRIX_HEIGHT];
00047     hardware::led_matrix ledMatrix_;
00052     hardware::joy_stick joyStick_;
00057     software::snake snake_;
00062     gameMode currentGameMode_{gameMode::normal};
00067     uint8_t numberOfFood_{0};
00068
00075     void setPixel(hardware::pixelCoordinate coord, pixelType type);
00083     bool isPixelFree(hardware::pixelCoordinate coord);
00091     bool isPixelFood(hardware::pixelCoordinate coord);
00099     bool isPixelObstacle(hardware::pixelCoordinate coord);
00107     bool isPixelSnake(hardware::pixelCoordinate coord);
00114     pixelType getPixelType(hardware::pixelCoordinate coord);
00120     void loseGame();
00129     void spawnFood(const uint8_t &number);
00142     bool makeMove();
00143
00144   public:
00153     game(hardware::led_matrix _ledMatrix, hardware::joy_stick _joyStick);
00158     ~game();
00169     void init();
00179     bool exec();
00187     bool menu();
00188 };
00189 }
00190
00191 #endif // __game_header_included__
```

4.2 joy_stick.hpp

```
00001 #ifndef __joy_stick_header_included__
00002 #define __joy_stick_header_included__
00003 #include <sys/_stdint.h>
00004 #include <Arduino.h>
00005
00006 namespace hardware {
00007
00008 class joy_stick {
00009   private:
00010     // pins
00016     const uint8_t buttonPin_;
00022     const uint8_t vrxPin_;
00028     const uint8_t vryPin_;
00029
00030     // resolution
00038     const uint16_t centerPoint_{2047};
00048     const uint16_t centerDetectRadius_{1000};
00049
00050     // setup
00056     bool initialized_{false};
00057
00068     bool isOutOfDetectBand(const int32_t &val);
00078     bool isOutOfDetectRadius(uint32_t x, uint32_t y);
00079
00080   public:
00088     joy_stick(uint8_t _buttonPin, uint8_t _vrxPin, uint8_t _vryPin);
00094     joy_stick(joy_stick &t);
00099     ~joy_stick();
00100
00105     enum direction {
00106       noDirection,
00107       up,
00108       down,
00109       left,
00110       right
00111     };
00112
00118     void init();
00124     direction getDirection();
00131     bool isPressed();
00132 };
00133 }
00134
00135 #endif // __joy_stick_header_included__
```

4.3 led_matrix.hpp

```
00001 #ifndef __led_matrix_header_included__
00002 #define __led_matrix_header_included__
00003
00004 #include <Adafruit_GFX.h>
00005 #include <FastLED_NeoMatrix.h>
00006 #include <FastLED.h>          // official FastLed 3.6.0 release doesnt work (missing uno r4 minima
      support)
00007                               // #1523 from facchinm (bit buggy but works) (04.12.2023)-->
      https://github.com/FastLED/FastLED/tree/0398b9a99901d00044de821ed86e8537995f561b

00008 #ifndef PSTR
00009  #define PSTR // Make Arduino Due happy
00010 #endif
00011
00012 #define COLOR_ORDER   GRB     //GRB     breaks if changed
00013 #define CHIPSET       WS2812B //WS2812B breaks if changed
00014 #define LED_PIN       5
00015 #define MATRIX_WIDTH  16
00016 #define MATRIX_HEIGHT 16
00017 #define NUM_LEDS      (MATRIX_HEIGHT * MATRIX_WIDTH)
00018
00019
00020 namespace hardware {
00021
00022 struct pixelCoordinate {
00023   uint8_t x_;
00024   uint8_t y_;
00031   bool increaseX();
00038   bool increaseY();
00045   bool decreaseX();
00052   bool decreaseY();
00053 };
00054
00055 inline bool operator==(const pixelCoordinate& first, const pixelCoordinate& second) {
```

```
00056   return (first.x_ == second.x_) && (first.y_ == second.y_);
00057 }
00058
00059 class led_matrix {
00060   protected:
00068     uint8_t brightness_{40};
00073     bool initailized_{false};
00078     CRGB leds_[NUM_LEDS];
00085     FastLED_NeoMatrix *matrix_;
00086
00087   public:
00092     led_matrix();
00098     led_matrix(led_matrix &t);
00103     ~led_matrix();
00108     enum colorPixel{
00109       obstacle = CRGB::Red,
00110       tail     = CRGB::Yellow,
00111       food     = CRGB::Green,
00112       head     = CRGB::Blue
00113     };
00119     void init();
00120
00121     //##### visualisation functions #####
00129     void setPixel(uint16_t x, uint16_t y, colorPixel color);
00137     void setPixel(uint16_t x, uint16_t y, uint32_t color);
00145     void unsetPixel(uint16_t x, uint16_t y);
00156     void printText(const char *text, int16_t x, int16_t y,
00157                    CRGB::HTMLColorCode color, bool wrap);
00163     void setBrightness(uint8_t brightn);
00168     void outputMatrix();
00174     void clearMatrix();
00175 };
00176 }
00177
00178 #endif // __led_matrix_header_included__
```

4.4 snake.hpp

```
00001 #ifndef __snake_header_included__
00002 #define __snake_header_included__
00003
00004 #include <deque>
00005 #include <algorithm>
00006 #include "led_matrix.hpp"
00007 #include "joy_stick.hpp"
00008
00009 namespace software {
00010 class snake {
00011   private:
00017     std::deque<hardware::pixelCoordinate> snake_;
00023     hardware::joy_stick::direction snakeDirection_;
00028     bool initialized_{false};
00036     bool contains(const hardware::pixelCoordinate &coord);
00037
00038   public:
00043     snake();
00048     ~snake();
00057     void init(hardware::pixelCoordinate coord,
00058               hardware::joy_stick::direction dir);
00065     void move(hardware::pixelCoordinate coord);
00072     void grow(hardware::pixelCoordinate coord);
00078     hardware::pixelCoordinate getHeadCoord();
00084     hardware::pixelCoordinate getTailCoord();
00090     hardware::joy_stick::direction getDirection();
00097     void setDirection(hardware::joy_stick::direction dir);
00104     uint8_t getSnakeLength();
00105 };
00106 }
00107
00108 #endif // __snake_header_included__
```

Index