

Snake játék soros portos irányítással

1.0

Készítette Doxygen 1.15.0

1. Osztálymutató	1
1.1. Osztálylista	1
2. Fájlmutató	3
2.1. Fájllista	3
3. Osztályok dokumentációja	5
3.1. Snake_body_s struktúreferencia	5
3.1.1. Részletes leírás	5
3.2. Snake_s struktúreferencia	5
3.2.1. Részletes leírás	6
4. Fájlok dokumentációja	7
4.1. main.c fájlreferencia	7
4.1.1. Részletes leírás	7
4.2. playfield.h fájlreferencia	7
4.2.1. Részletes leírás	7
4.3. playfield.h	8
4.4. points.h fájlreferencia	8
4.4.1. Részletes leírás	8
4.4.2. Függvények dokumentációja	8
4.4.2.1. gameOverLED()	8
4.5. points.h	9
4.6. snake.h fájlreferencia	9
4.6.1. Részletes leírás	10
4.6.2. Enumerációk dokumentációja	10
4.6.2.1. Dir_e	10
4.7. snake.h	11
4.8. usart.h	11
Tárgymutató	13

1. fejezet

Osztálymutató

1.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

Snake_body_s	A struktúra, amely tárolja a kígyó minden darabkáját	5
Snake_s	A kígyó struktúrája	5

2. fejezet

Fájlmutató

2.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

main.c	Inicializálja a hardver perifériáit, a kígyót és folyamatosan futtatja a gameTick()-t	7
playfield.h	Inicializálja az LCD kijelzőt	7
points.h	A pontszámláló segéd tömbje és kezelő függvényei	8
snake.h	A kígyó struktúrái és annak kezelő függvényeinek fejléce	9
usart.h	11

3. fejezet

Osztályok dokumentációja

3.1. Snake_body_s struktúrareferencia

A struktúra, amely tárolja a kígyó minden darabkáját.

```
#include <snake.h>
```

Publikus attribútumok

- uint8_t **digitNum**
- SegmentLCD_LowerCharSegments_TypeDef **snakeBody**

3.1.1. Részletes leírás

A struktúra, amely tárolja a kígyó minden darabkáját.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- [snake.h](#)

3.2. Snake_s struktúrareferencia

A kígyó struktúrája.

```
#include <snake.h>
```

Publikus attribútumok

- [Dir_e](#) **dir**
- [Snake_body_s](#) **snakePart** [[SNAKE_MAX_LENGTH](#)]
- SegmentLCD_LowerCharSegments_TypeDef **head** [7]
- SegmentLCD_LowerCharSegments_TypeDef **end** [7]
- uint8_t **length**

3.2.1. Részletes leírás

A kígyó struktúrája.

Megjegyzés

Tárolja a kígyó irányát, a kígyó "darabkái", a kígyó fejének pozícióját és a hosszát.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- [snake.h](#)

4. fejezet

Fájlok dokumentációja

4.1. main.c fájlreferencia

Inicializálja a hardver perifériáit, a kígyót és folyamatosan futtatja a gameTick()-t.

```
#include "em_device.h"
#include "em_gpio.h"
#include "stdio.h"
#include "usart.h"
#include "snake.h"
#include "game.h"
```

Függvények

- int **main** (void)

4.1.1. Részletes leírás

Inicializálja a hardver perifériáit, a kígyót és folyamatosan futtatja a gameTick()-t.

4.2. playfield.h fájlreferencia

Inicializálja az LCD kijelzőt.

Függvények

- void **initPlayfield** (void)
Inicializál, majd törli az LCD kijelző tartalmát.

4.2.1. Részletes leírás

Inicializálja az LCD kijelzőt.

4.3. playfield.h

[Ugrás a fájl dokumentációjához.](#)

```
00001
00005
00006 #ifndef PLAYFIELD_H
00007 #define PLAYFIELD_H
00008
00009 void initPlayfield(void);
00010
00011 #endif
```

4.4. points.h fájlreferencia

A pontszámláló segéd tömbje és kezelő függvényei.

```
#include "segmentlcdconfig.h"
#include "segmentlcd.h"
#include "segmentlcd_individual.h"
```

Függvények

- void **gameOverLED** (int state)
A villogtatás megvalósítása.
- void **gameOverBlink** (void)
Ez fogja villogtatás alapját adni.
- void **showPoints** (void)
Kijelzi, hogy jelenleg milyen hosszú a kígyó.

4.4.1. Részletes leírás

A pontszámláló segéd tömbje és kezelő függvényei.

4.4.2. Függvények dokumentációja

4.4.2.1. gameOverLED()

```
void gameOverLED (
    int state)
```

A villogtatás megvalósítása.

Paraméterek

<i>state</i>	Milyen állapotban vannak épp a pontok.
--------------	--

4.5. points.h

[Ugrás a fájl dokumentációjához.](#)

```
00001
00005 #ifndef POINTS_H
00006 #define POINTS_H
00007
00008 #include "segmentlcdconfig.h"
00009 #include "segmentlcd.h"
00010 #include "segmentlcd_individual.h"
00011
00015 static const lcdSymbol gameOverLEDs[] = {
00016     LCD_SYMBOL_DP2,
00017     LCD_SYMBOL_DP3,
00018     LCD_SYMBOL_DP4,
00019     LCD_SYMBOL_DP5,
00020     LCD_SYMBOL_DP6
00021 };
00022
00026 static const int gameOverLEDCount = sizeof(gameOverLEDs)/sizeof(gameOverLEDs[0]);
00027
00028 void gameOverLED(int state);
00029
00030 void gameOverBlink(void);
00031
00032 void showPoints(void);
00033
00034 #endif
```

4.6. snake.h fájlreferencia

A kígyó struktúrái és annak kezelő függvényeinek fejléce.

```
#include "segmentlcd.h"
#include "segmentlcd_individual.h"
#include <stdint.h>
```

Osztályok

- struct [Snake_body_s](#)
A struktúra, amely tárolja a kígyó minden darabkáját.
- struct [Snake_s](#)
A kígyó struktúrája.

Makródefiníciók

- #define **SNAKE_MAX_LENGTH** 50
A kígyó max hossza. Ilyen hosszú valóban nem lehet.

Enumerációk

- enum [Dir_e](#) { [UP](#) , [DOWN](#) , [LEFT](#) , [RIGHT](#) }
A képernyőn milyen irányba halad a kígyó.

Függvények

- void **initSnake** (void)
Inicializálja a kígyót, generál egy ételt és kirajzolja ezeket.
- void **generateFood** (void)
Random generál a pályán egy ételt.
- void **drawFoodAndSnake** (void)
Összemossa a kígyó helyét és az étel helyét a pályán, majd kirajzolja egyben.
- void **moveSnake** (void)
A kígyó mozgását biztosítja. Tudja kezelni, hogyha irányt változtat.
- void **updateSnake** (void)
Frissíti a kígyó helyzetét. Teszteli, hogy evett-e ételt.
- void **generateCurrentState** (void)
Ez függvény mossa össze a kígyó testét és az étel helyét egy tömbbe.
- void **updateDirection** (void)
Ha új karakter érkezett akkor frissíti, hogy milyen irányba halad a kígyó.
- bool **checkFood** (void)
Ellenőrzi, hogy a kígyó megevett-e egy ételt.
- bool **checkCollision** (void)
Ellenőrzi, hogy magába ütközött-e.

Változók

- **Snake_s snake**
A kígyó példánya.

4.6.1. Részletes leírás

A kígyó struktúrái és annak kezelő függvényeinek fejléce.

4.6.2. Enumerációk dokumentációja

4.6.2.1. Dir_e

enum **Dir_e**

A képernyőn milyen irányba halad a kígyó.

Enumeráció-értékek

UP	Fel
DOWN	Le
LEFT	Balra
RIGHT	Jobbra

4.7. snake.h

[Ugrás a fájl dokumentációjához.](#)

```

00001
00005
00006 #ifndef SNAKE_H
00007 #define SNAKE_H
00008
00009 #include "segmentlcd.h"
00010 #include "segmentlcd_individual.h"
00011 #include <stdint.h>
00012
00013
00014 #define SNAKE_MAX_LENGTH 50
00015
00019 typedef enum {
00020     UP,
00021     DOWN,
00022     LEFT,
00023     RIGHT
00024 }Dir_e;
00025
00029 typedef struct {
00030     uint8_t digitNum;
00031     SegmentLCD_LowerCharSegments_TypeDef snakeBody;
00032 }Snake_body_s;
00033
00038 typedef struct {
00039     Dir_e dir; //a kígyó jelenlegi iránya
00040     Snake_body_s snakePart[SNAKE_MAX_LENGTH];
00041     SegmentLCD_LowerCharSegments_TypeDef head[7]; //a kígyó feje
00042     SegmentLCD_LowerCharSegments_TypeDef end[7]; // a kígyó testének vége
00043     uint8_t length; //jelenleg milyen hosszú
00044 }Snake_s;
00045
00046 extern Snake_s snake;
00047
00048
00049 void initSnake(void);
00050 void generateFood(void);
00051 void drawFoodAndSnake(void);
00052 void moveSnake(void);
00053 void updateSnake(void);
00054
00055
00056 void generateCurrentState(void);
00057 void updateDirection(void);
00058 bool checkFood(void);
00059 bool checkCollision(void);
00060
00061 #endif

```

4.8. usart.h

```

00001
00005
00006
00007
00008 #ifndef USART_H
00009 #define USART_H
00010
00011 extern volatile char usartValue;
00012 extern volatile bool progressed;
00013
00014 void usartInit(void);
00015 void UART0_RX_IRQHandler(void);
00016 void GetChar(void);
00017
00018 #endif

```


Tárgymutató

Dir_e
 snake.h, [10](#)
DOWN
 snake.h, [10](#)

gameOverLED
 points.h, [8](#)

LEFT
 snake.h, [10](#)

main.c, [7](#)

playfield.h, [7](#)
points.h, [8](#)
 gameOverLED, [8](#)

RIGHT
 snake.h, [10](#)

snake.h, [9](#)
 Dir_e, [10](#)
 DOWN, [10](#)
 LEFT, [10](#)
 RIGHT, [10](#)
 UP, [10](#)
Snake_body_s, [5](#)
Snake_s, [5](#)

UP
 snake.h, [10](#)