Оглавление

[Добавление новых состояний. 1](#_Toc122888301)

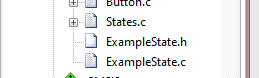
[Как изменить состояние из другого состояния? 6](#_Toc122888302)

[Что такое StateData? 6](#_Toc122888303)

# Добавление новых состояний.

Чтобы добавить новое состояние необходимо придерживаться нескольких правил. Для наглядности добавим новое состояние, описывая каждый шаг.

Для начала создадим новые файлы **.h/.c.**



Лучшей практикой будет считаться создание состояний с однозначным названием, а также припиской **State.** Например: MovementState, MenuState.

Затем заполним файл **.h** следующим образом.

#ifndef EXAMPLESTATE\_H

#define EXAMPLESTATE\_H

#include "States.h"

void example\_state\_update(StateData\*);

void example\_state\_start(StateData\*);

void example\_state\_changer(void (\*change\_method)(int));

void example\_state\_input(uint32\_t code, StateData\*);

#endif

Для повышения читаемости все методы имеют однозначный нейминг и всегда содержат в себе название состояния. Предоставленные функции являются обязательными, но и расширять другими можно без проблем.

Теперь подробнее о методах:

**example\_state\_update(StateData\*) –** этот метод выполняется с таймингом, указанным в **main.c.** В нем можно делать различные анимации или обрабатывать любую логику, соответствующую состоянию.

**example\_state\_start(StateData\*) –** этот метод выполняется при смене состояния самым первым. Необходим для начальной инициализации объектов состояния, например здесь можно обнулить счетчик очков.

**example\_state\_changer(void (\*change\_method)(int)) –** этот метод необходим для установки функции смены состояния на другое. То есть внутри этого состояния мы всегда можем вызвать метод, что указывает на главную функцию изменения состояния в системе **StateSystem.c.**

**example\_state\_input(uint32\_t code, StateData\*) –** этот метод занимается обработкой нажатий на кейпаде. Информацию он получает от **StateSystem.**

Теперь опишем файл **ExampleState.c**

#include "ExampleState.h"

void (\*example\_state\_change) (int);

void example\_state\_update(StateData\* state\_data)

{

}

void example\_state\_start(StateData\* state\_data)

{

}

void example\_state\_changer(void (\*change\_method)(int))

{

example\_state\_change = change\_method;

}

void example\_state\_input(uint32\_t code, StateData\* state\_data)

{

}

Файл всегда должен определять эти функции. Особое внимание стоит уделить указателю на функцию изменения состояния. Он всегда должен присутствовать в этом файле, а метод **example\_state\_changer(void (\*change\_method)(int))** всегда должен определяться именно так, чтобы установить ссылку на функцию **StateSystem.**

Теперь рассмотрим файл **StateSystem.h.**

#ifndef STATESYSTEM\_H

#define STATESYSTEM\_H

#include "MenuState.h"

#include "SnakeMovementState.h"

#include "GameOverState.h"

#include "ExampleState.h"

#include "States.h"

void state\_system\_init(void);

void state\_system\_update(void);

void state\_system\_input(uint32\_t code);

void change\_state(int state\_id);

#endif

Здесь уже подключены несколько состояний. Добавим новое – **ExampleState.h.**

Рассмотрим **StateSystem.c.**

#include "StateSystem.h"

State\*\* states;

int CURRENT\_STATE;

StateData\* state\_data;

void state\_system\_init(void)

{

int states\_count = 4;

states = (State\*\*) malloc(sizeof(State\*\*) \* states\_count);

states[0] = CreateState(snake\_movement\_state, snake\_movement\_state\_start, snake\_movement\_state\_changer, snake\_movement\_state\_input);

states[1] = CreateState(menu\_state, menu\_state\_start, menu\_state\_changer, menu\_state\_input);

states[2] = CreateState(g\_over\_state, g\_over\_state\_start, g\_over\_state\_changer, g\_over\_state\_input);

states[3] = CreateState(example\_state\_update, example\_state\_start, example\_state\_changer, example\_state\_input);

state\_data = NEW\_DATA;

state\_data->score = 0;

state\_data->best\_score = 0;

state\_data->level = 0;

change\_state(MENU\_STATE);

}

void state\_system\_update(void)

{

states[CURRENT\_STATE]->update(state\_data);

printf("state\_update\n");

}

void change\_state(int state\_id)

{

printf("state\_changed\n");

CURRENT\_STATE = state\_id;

states[CURRENT\_STATE]->start(state\_data);

states[CURRENT\_STATE]->changer(change\_state);

}

void state\_system\_input(uint32\_t code)

{

printf("state\_input\n");

states[CURRENT\_STATE]->input(code, state\_data);

}

Чтобы система знала, какое состояние мы добавили, необходимо в функции **state\_system\_init** увеличить число **states\_count** и прописать следующую строку:

states[3] = CreateState(example\_state\_update, example\_state\_start, example\_state\_changer, example\_state\_input);

Метод **CreateState** выделяет память для структуры **State** и инициализирует ее. Первым аргументом всегда идет функция обновления состояния, вторым – инициализации состояния, третьим – изменения, четвертым – обработчика нажатий.

Теперь рассмотрим **States.h**

#ifndef STATES\_H

#define STATES\_H

/\*\*

@file

@brief ñîñòîÿíèÿ èãðû

\*/

#include <cstdlib>

#include <stdint.h>

#define SNAKE\_MOVEMENT\_STATE 0

#define MENU\_STATE 1

#define GAME\_OVER\_STATE 2

#define EXAMPLE\_STATE 3

#define NEW\_DATA ((StateData\*) malloc((sizeof(StateData))))

typedef struct {

int score;

int best\_score;

int level;

} StateData;

typedef struct {

void (\*update) (StateData\*);

void (\*start) (StateData\*);

void (\*changer) ();

void (\*input) (uint32\_t, StateData\*);

} State;

extern State\* CreateState(void (\*update) (StateData\*), void (\*start) (StateData\*), void (\*changer) (), void (\*input) (uint32\_t, StateData\*));

#endif

Для того чтобы понимать какое состояние передать в систему для изменения, необходимо прописать его здесь. Добавим строчку

**#define EXAMPLE\_STATE 3**

Она указывает на индекс только что созданного состояния.

**Состояние добавлено.**

# Как изменить состояние из другого состояния?

Чтобы изменить состояние, например из **ExampleState**, необходимо вызвать метод изменения состояния в месте нужном вам. Если вы хотите изменить состояние на меню при нажатии любой кнопки, то нужно прописать это в соответствующем методе следующим образом:

void example\_state\_input(uint32\_t code, StateData\* state\_data)

{

example\_state\_change(MENU\_STATE);

}

Если все сделано верно, то при нажатии любой кнопки отрисуется меню.

# Что такое StateData?

Данная структура нужна для обмена любыми данными между состояниями. Например в игре «Змейка» она использовалась для передачи счета игрока, чтобы отображать его там, где это необходимо.

typedef struct {

int score;

int best\_score;

int level;

} StateData;

Вы можете добавить сюда все, что угодно. А пользоваться этим будет очень удобно.