

The background is a vibrant, abstract composition of various colored circles and organic shapes. A large orange circle is the central focus, containing the text. Surrounding it are several other shapes: a large yellow circle in the top left, a blue circle in the top right, a green circle in the bottom right, and a blue circle in the bottom left. There are also smaller circles in blue, black, and grey scattered throughout the design. The overall aesthetic is playful and modern.

Fun with Allegro

# Bool data type

- A type can only be assigned by two values
  - 0 – means false.
  - 1 – means true.
  - You can just assign true or false to a bool variable.

(eg: `bool HT_good = true;`  
`bool a = false;`)

- A data type that in lots of programming language
  - Java, C++ and so on
- Allegro implements a bool data type in itself



# Outline

**01**

Introduction

**02**

Draw something!

**03**

Deal with event !

**04**

Play some music!

**05**

Supplementary



# Outline

01

Introduction

02

Draw something!

03

Deal with event !

04

Play some music!

05

Supplementary

# Hello to Allegro

- The functions of allegro

```
#define al_init() (al_install_system(ALLEGRO_VERSION_INT, atexit))
ALLEGRO_DISPLAY* al_create_display(int width, int height);
ALLEGRO_COLOR al_map_rgb(unsigned char r, unsigned char g, unsigned char b);
void al_set_window_position(ALLEGRO_DISPLAY *display, int x, int y)
void al_clear_to_color(ALLEGRO_COLOR color);
void al_flip_display(void);
void al_destroy_display(ALLEGRO_DISPLAY *display);
void al_rest(double seconds);
```

# Hello to Allegro

- The functions of allegro

```
#define al_init() (al_install_system(ALLEGRO_VERSION_INT, atexit))
ALLEGRO_DISPLAY* al_create_display(int width, int height);
ALLEGRO_COLOR al_map_rgb(unsigned char r, unsigned char g, unsigned char b);
void al_set_window_position(ALLEGRO_DISPLAY *display, int x, int y)
void al_clear_to_color(ALLEGRO_COLOR color);
void al_flip_display(void);
void al_destroy_display(ALLEGRO_DISPLAY *display);
void al_rest(double seconds);
```

→ A define of other function that return "Bool" type

# Hello to Allegro

- The functions of allegro

```
#define al_init() (al_install_system(ALLEGRO_VERSION_INT, atexit))
ALLEGRO_DISPLAY* al_create_display(int width, int height);
ALLEGRO_COLOR al_map_rgb(unsigned char r, unsigned char g, unsigned char b);
void al_set_window_position(ALLEGRO_DISPLAY *display, int x, int y)
void al_clear_to_color(ALLEGRO_COLOR color);
void al_flip_display(void);
void al_destroy_display(ALLEGRO_DISPLAY *display);
void al_rest(double seconds);
```

→ A function that return "the pointer of ALLEGRO\_DISPLAY" type

# Hello to Allegro

- The functions of allegro

```
#define al_init() (al_install_system(ALLEGRO_VERSION_INT, atexit))
ALLEGRO_DISPLAY* al_create_display(int width, int height);
ALLEGRO_COLOR al_map_rgb(unsigned char r, unsigned char g, unsigned char b);
void al_set_window_position(ALLEGRO_DISPLAY *display, int x, int y)
void al_clear_to_color(ALLEGRO_COLOR color);
void al_flip_display(void);
void al_destroy_display(ALLEGRO_DISPLAY *display);
void al_rest(double seconds);
```

→ A function that return "ALLEGRO\_COLOR" type



# Hello to Allegro

- The basic componenet – display a window!

include

Init

Display

Destroy

# Hello to Allegro

- The basic componenet – display a window!

include

Init

Display

Destroy

```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

Include the function in <allegro5/allegro.h>.

# Hello to Allegro

- The basic componenet – display a window!

include

Init

Display

Destroy

```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

```
void game_init() {
    al_init()
    display = al_create_display(width, height);
    al_set_window_position(display, 0, 0);
}
```

Initialize the function in  
<allegro5/allegro.h>.

# Hello to Allegro

- The basic componenet – display a window!

include

Init

Display

Destroy

```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

```
void game_init() {
    al_init()
    display = al_create_display(width, height);
    al_set_window_position(display, 0, 0);
}
```

Set the window  
with width and  
height

# Hello to Allegro

- The basic componenet – display a window!

include

Init

Display

Destroy

```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

```
void game_init() {
    al_init()
    display = al_create_display(width, height);
    al_set_window_position(display, 0, 0);
}
```

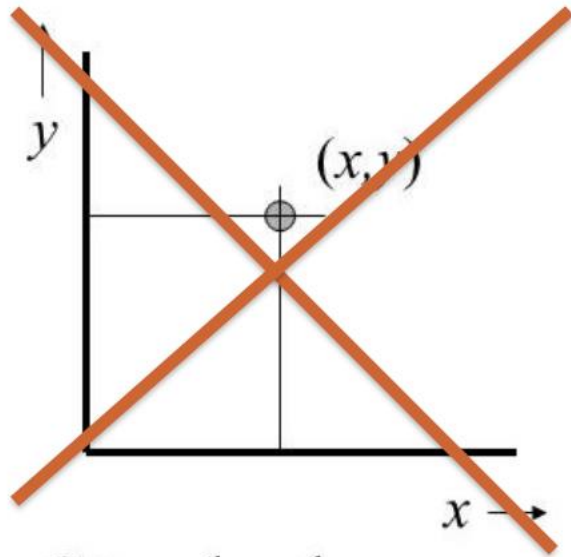
Set the window  
with position

# Hello to Allegro

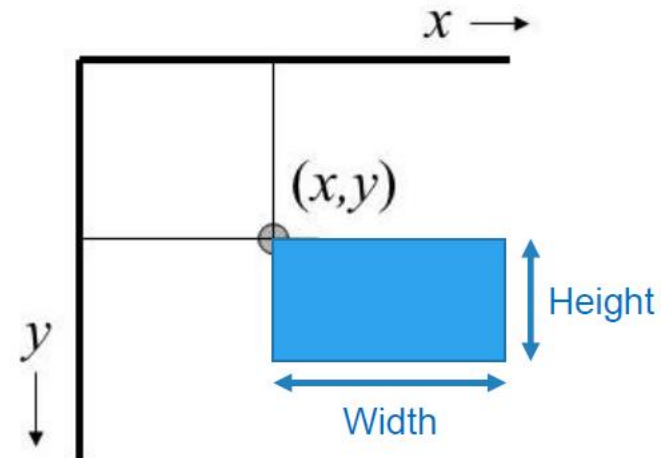
- The basic componenet – display a window!



Position of window



~~Standard~~



Screen (output, input)

# Hello to Allegro

- The basic componenet – display a window!

include

Init

Display

Destroy

```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

```
void game_begin() {
    al_clear_to_color( al_map_rgb(255, 0, 0) );
    al_flip_display();
}
```

Fill the window  
with red

# Hello to Allegro

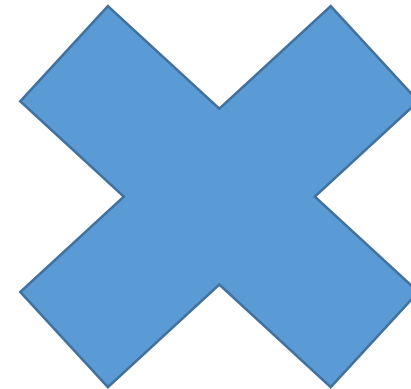
- The basic component – display a window!



Buffer



Not display yet!





# Hello to Allegro

- The basic componenet – display a window!

include

Init

Display

Destroy

```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

```
void game_begin() {
    al_clear_to_color( al_map_rgb(255, 0, 0) );
    al_flip_display();
}
```

Display the result

# Hello to Allegro

- The basic component – display a window!



Buffer



`al_flip_display()`



# Hello to Allegro

- The basic component – display a window!



```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

Stop the program for 5 seconds so  
the window you create will stay  
for 5 seconds

# Hello to Allegro

- The basic component – display a window!

include

Init

Display

Destroy

```
#include <stdio.h>
#include <allegro5/allegro.h>
ALLEGRO_DISPLAY* display = NULL;
int main(int argc, char *argv[]) {
    game_init();
    game_begin();
    al_rest(5);
    game_destroy();
    return 0;
}
```

```
void game_destroy() {
    al_destroy_display(display);
}
```

Release the memory of  
the variable

The background is a vibrant, abstract composition. It features a large, irregular white shape in the center, surrounded by various colored organic shapes. There are large grey circles in the top-left and bottom-right, a large blue shape in the top-center, a large yellow pill-shaped shape on the right, and a large orange circle in the bottom-right. Smaller circles and dots in shades of blue, black, and light green are scattered throughout the composition.

# Practice

Finish task 1!!



# Outline

01

Introduction

02

Draw something!

03

Deal with event !

04

Play some music!

05

Supplementary

# Draw text

- Draw some text on the window



# Draw text

- Draw some text on the window



```
#include <allegro5/allegro_font.h>  
#include <allegro5/allegro_ttf.h>
```



Head file of font and ttf(字體)



# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_init() {  
    al_init()  
    display = al_create_display(width, height);  
    al_set_window_position(display, 0, 0);  
    al_init_font_addon();  
    al_init_ttf_addon();  
}
```

initialize file of font and ttf(字體)

# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255,255,255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

Load the font

# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255, 255, 255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

→ The path of the font file

# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255, 255, 255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

→ The size of the font

# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255,255,255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

Usually 0, but you can put other two flags:

1. ALLEGRO\_TTF\_NO\_KERNING: not use any kerning(間距調整)
2. ALLEGRO\_TTF\_MONOCHROME: Load as a monochrome font (Which means no anti-aliasing of the font is done)
3. Eg: `al_load_ttf_font("pirulen.ttf", 50, ALLEGRO_TTF_NO_KERNING);`

# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255,255,255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

→ Draw text into buffer

# Draw text

- Draw some text on the window

Include

Init

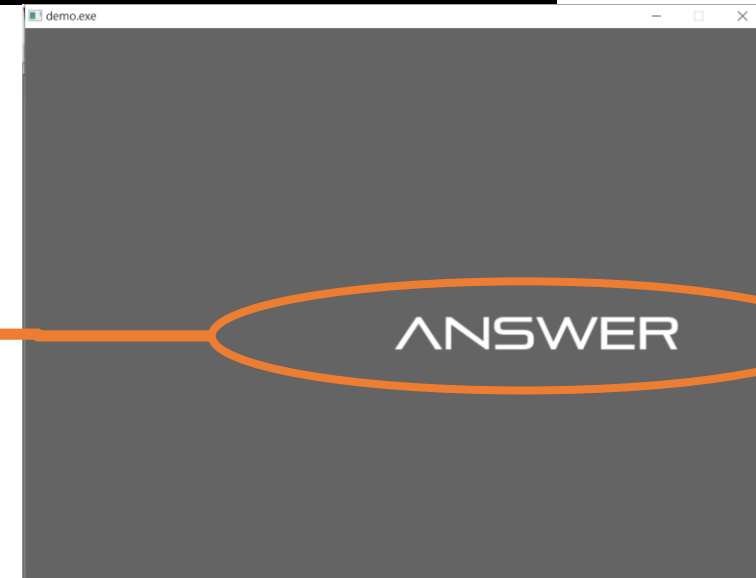
Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255, 255, 255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

Set the color of text

Color white



# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255, 255, 255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

The position of the text

Position (width/2, height/2)





# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255, 255, 255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

You can put three different flags here:

1. ALLEGRO\_ALIGN\_LEFT: text align-left( you can use 0 instead )
2. ALLEGRO\_ALIGN\_CENTRE: text align-center
3. ALLEGRO\_ALIGN\_RIGHT: text align-right

# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

Text align-right



Text align-center



Text align-left



# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_begin() {  
    al_clear_to_color( al_map_rgb(100, 100, 100) );  
    font = al_load_ttf_font("pirulen.ttf", 50, 0);  
    al_draw_text(font, al_map_rgb(255,255,255), width/2, height/2, 0, "answer");  
    al_flip_display();  
}
```

The text you want to show

# Draw text

- Draw some text on the window

Include

Init

Display

Destroy

```
void game_destroy() {  
    al_destroy_display(display);  
    al_destroy_font(font);  
}
```

Release the memory of font

# Draw rectangle

- Draw rectangle on the window



# Draw rectangle

- Draw rectangle on the window

Include

Init

Display

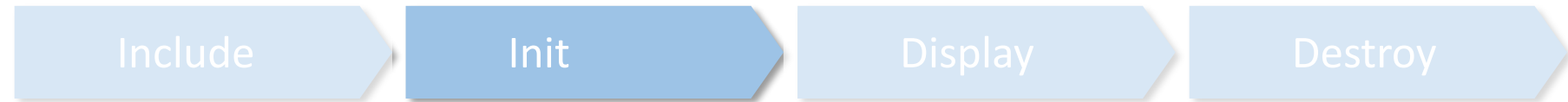
Destroy

```
#include <allegro5/allegro_primitives.h>
```

Include the head file can draw some basic things

# Draw rectangle

- Draw rectangle on the window



```
al_init_primitives_addon();
```



Initialize primitive functions

# Draw rectangle

- Draw rectangle on the window

Include

Init

Display

Destroy

```
al_draw_rectangle(width-300, height-50, width, height, al_map_rgb(255,255,255), 2);
```

Draw a hollow rectangle



# Draw rectangle

- Draw rectangle on the window

Include

Init

Display

Destroy

```
al_draw_rectangle(width-300, height-50, width, height, al_map_rgb(255,255,255), 2);
```

The position of the up left corner of rectangle



# Draw rectangle

- Draw rectangle on the window

Include

Init

Display

Destroy

```
al_draw_rectangle(width-300, height-50, width, height, al_map_rgb(255,255,255), 2);
```

The position of the down right corner of rectangle



# Draw rectangle

- Draw rectangle on the window

Include

Init

Display

Destroy

```
al_draw_rectangle(width-300, height-50, width, height, al_map_rgb(255,255,255), 2);
```

The color of the rectangle

Color white



# Draw rectangle

- Draw rectangle on the window

Include

Init

Display

Destroy

```
al_draw_rectangle(width-300, height-50, width, height, al_map_rgb(255,255,255), 2);
```

The width of edges of the rectangle

The width of edges of the rectangle is 2

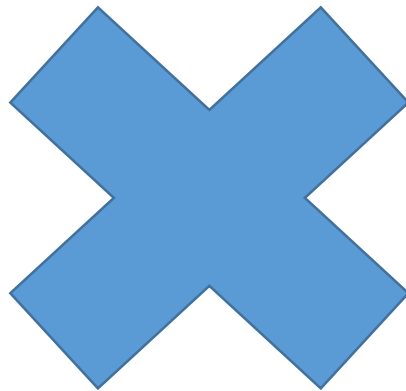


# Draw rectangle

- Draw rectangle on the window



No need to destroy



# Draw picture

- Draw picture on the window



# Draw picture

- Draw picture on the window



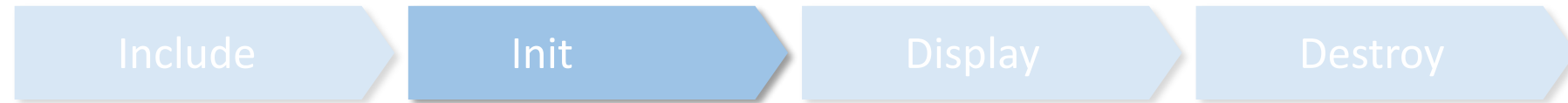
```
#include <allegro5/allegro_image.h>
```



Include the head file can draw picture on the window

# Draw picture

- Draw picture on the window



```
al_init_image_addon();
```

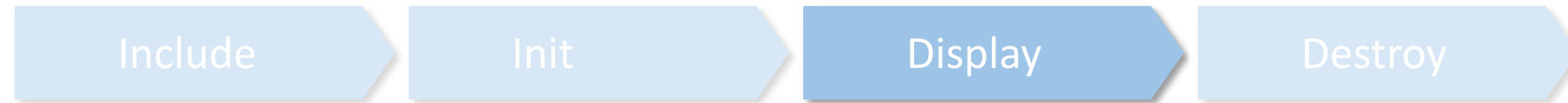


Initialize the image function

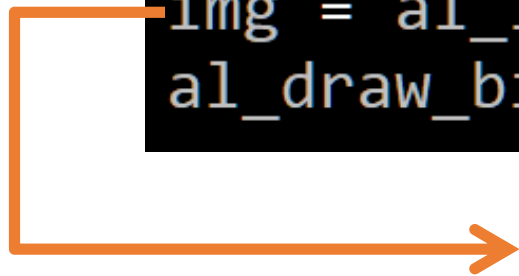


# Draw picture

- Draw picture on the window

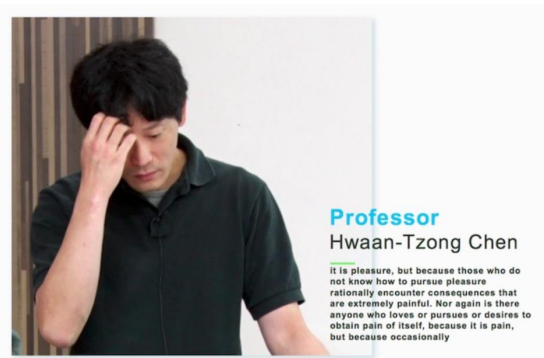


```
img = al_load_bitmap("htchen.jpg");  
al_draw_bitmap(img, 0, 0, 0);
```



Store the image in to variable "img"

img:



# Draw picture

- Draw picture on the window

Include

Init

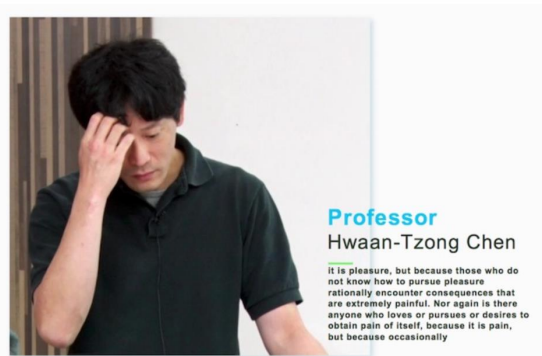
Display

Destroy

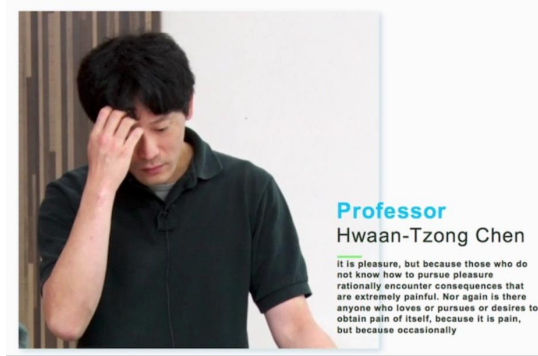
```
img = al_load_bitmap("htchen.jpg");  
al_draw_bitmap(img, 0, 0, 0);
```

Draw the image into buffer

img:



buffer:



Use `al_flip_display()`,  
to show the image  
on the window

# Draw picture

- Draw picture on the window

Include

Init

Display

Destroy

```
img = al_load_bitmap("htchen.jpg");  
al_draw_bitmap(img, 0, 0, 0);
```

The position of the up left corner of the image



# Draw picture

- Draw picture on the window



```
img = al_load_bitmap("htchen.jpg");  
al_draw_bitmap(img, 0, 0, 0);
```

Usually 0, but you can put other two flags:

1. ALLEGRO\_FLIP\_HORIZONTAL: flip the image horizontally
2. ALLEGRO\_FLIP\_VERTICAL: flip the image vertically



# Draw picture

- Draw picture on the window

Include

Init

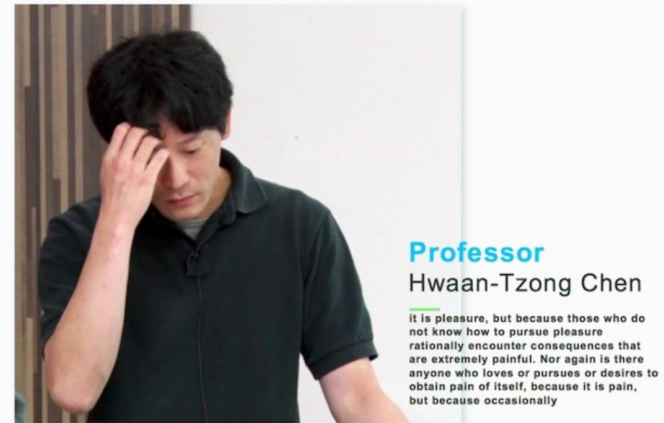
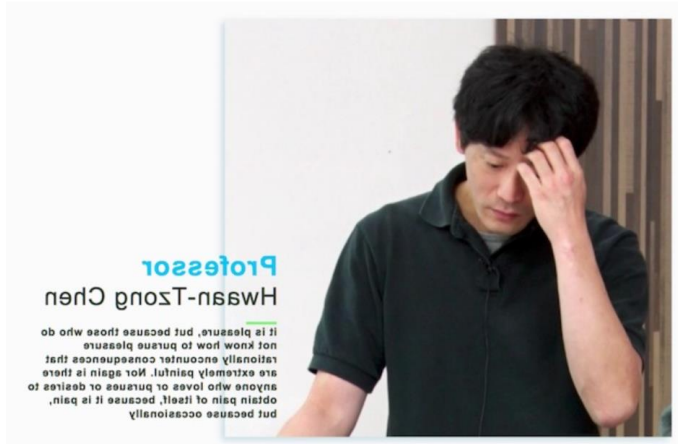
Display

Destroy

ALLEGRO\_FLIP\_HORIZONTAL

0

ALLEGRO\_FLIP\_VERTICAL



# Draw picture

- Draw picture on the window



```
al_destroy_bitmap(img);
```



Release the memory of the image

The background is a vibrant, abstract composition. It features a large, irregular white shape in the center, surrounded by various colored organic shapes. These include a large grey circle in the top left, a large blue shape in the top center, a large yellow pill-shaped shape on the right, a large orange circle in the bottom right, and several smaller circles in shades of blue, green, and black scattered throughout. The overall style is modern and playful.

# Practice

Finish task 2!!



# Outline

01

Introduction

02

Draw something!

03

Deal with event !

04

Play some music!

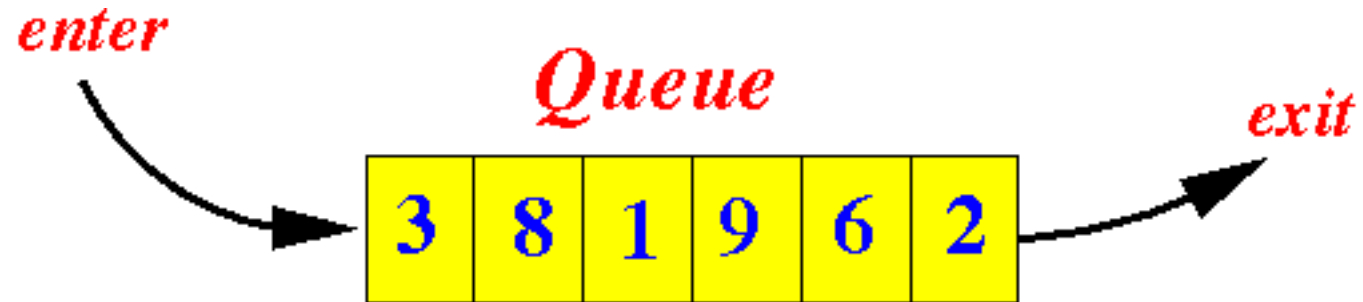
05

Supplementary

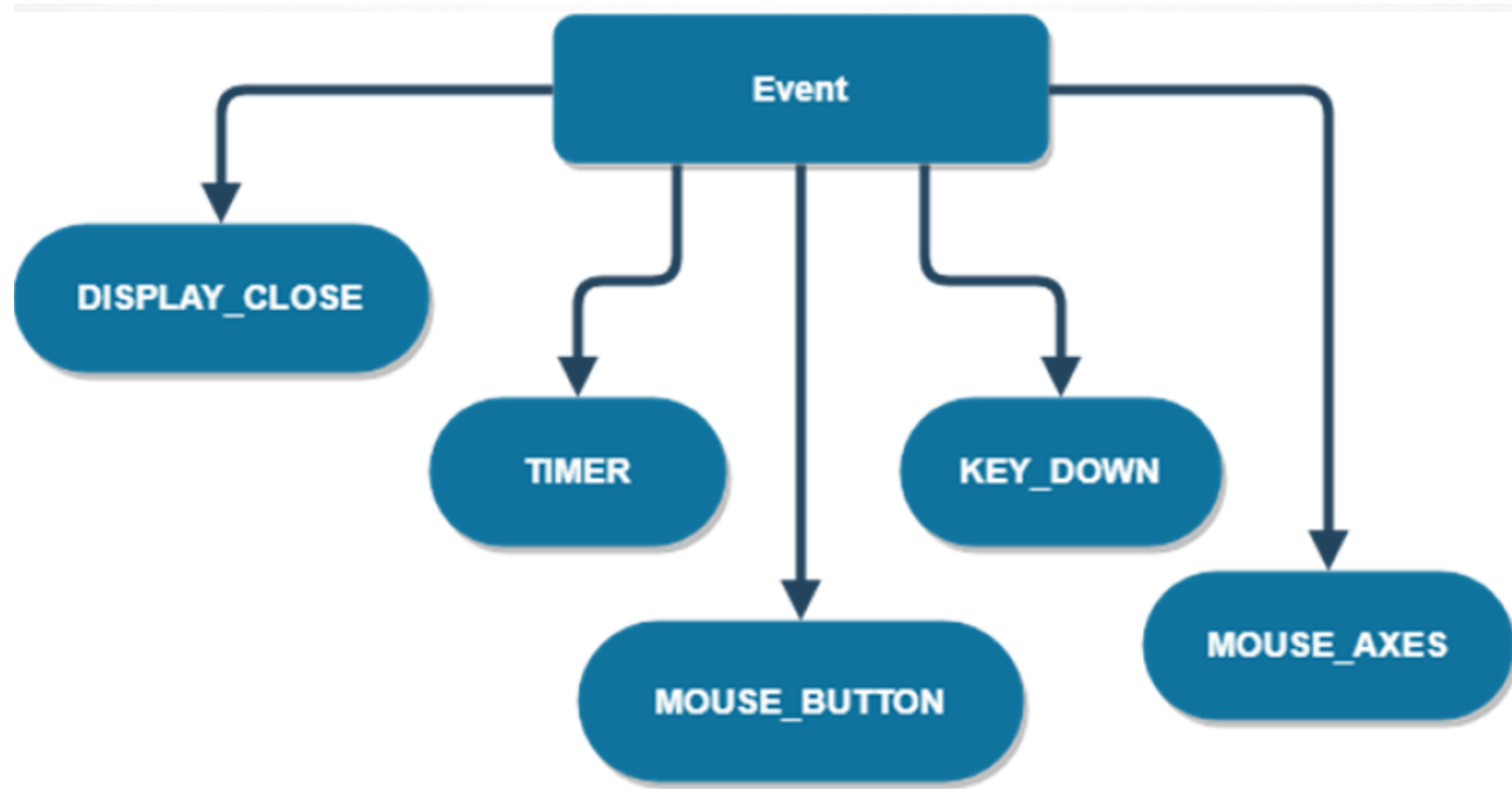


# queue

The one that enters the queue first will also exit first (because he/she gets serviced first)

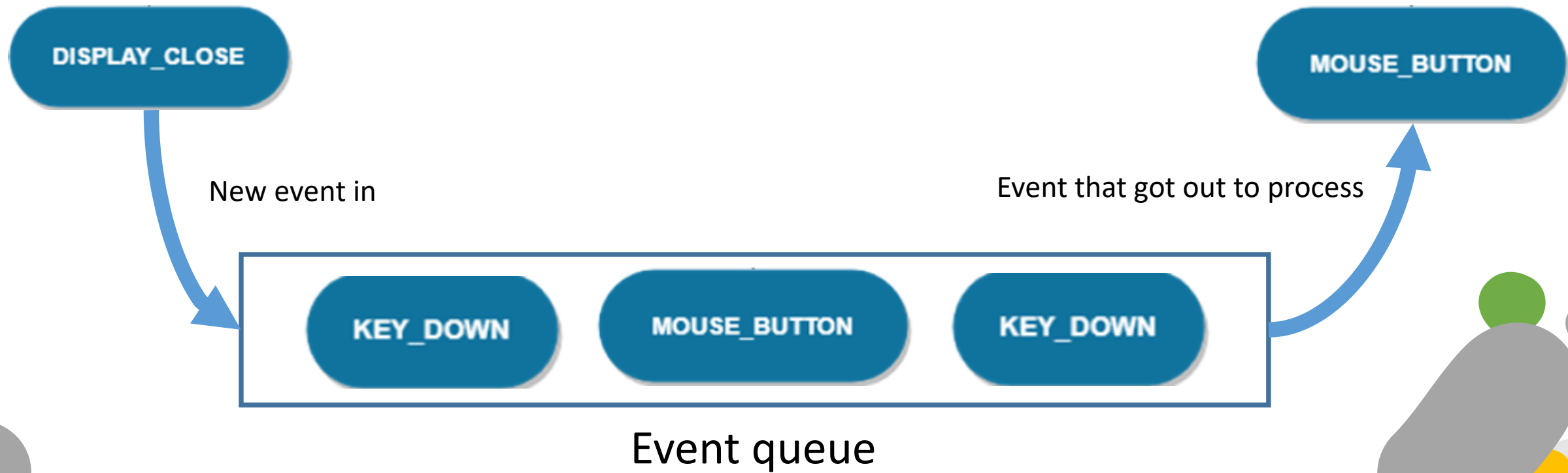


# Type of event



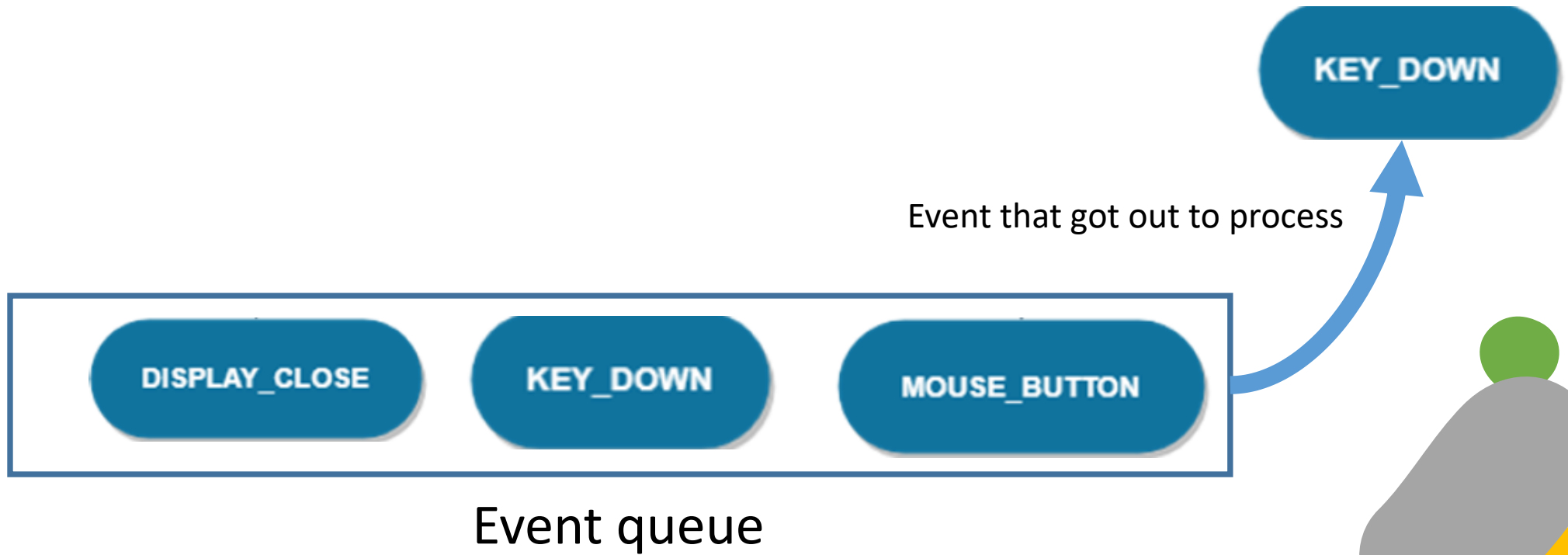
# Event queue

A queue that store event



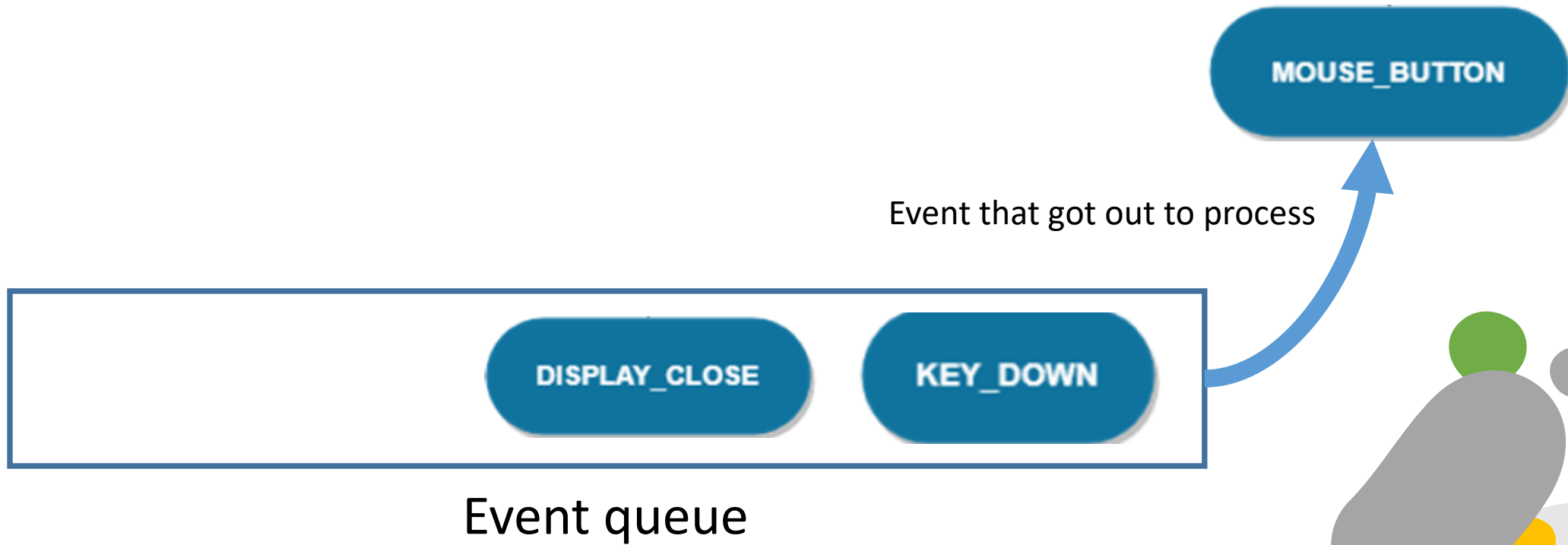
# Event queue

A queue that store event



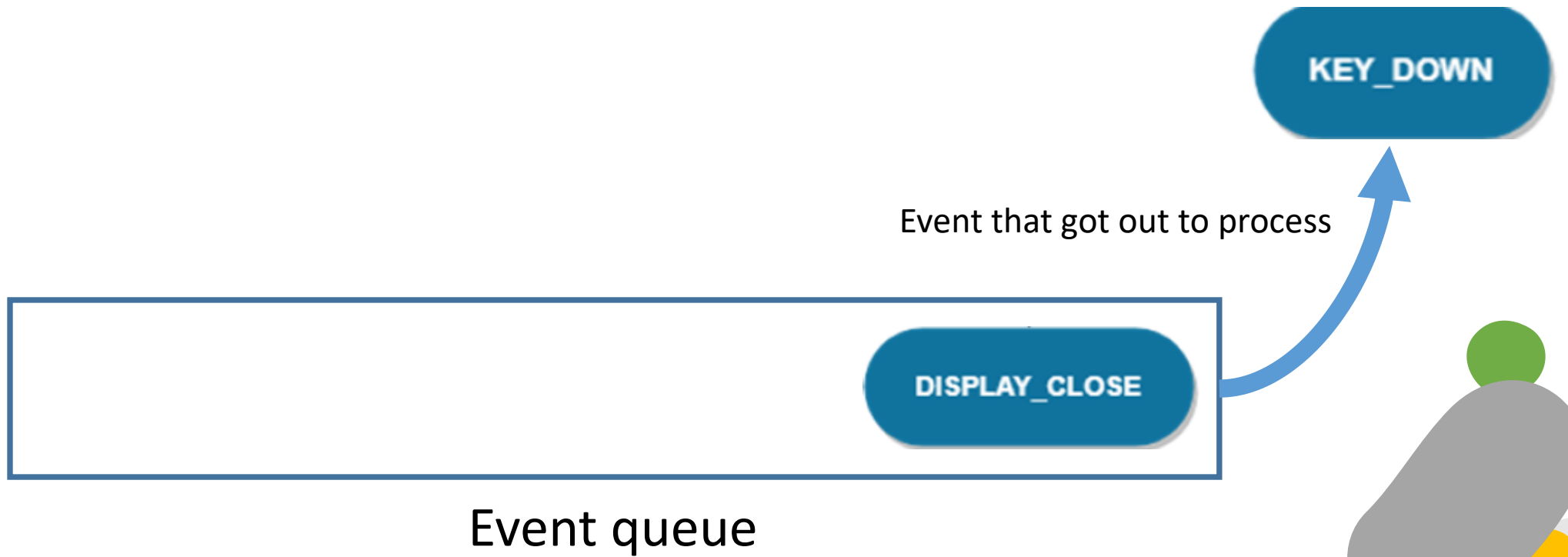
# Event queue

A queue that store event



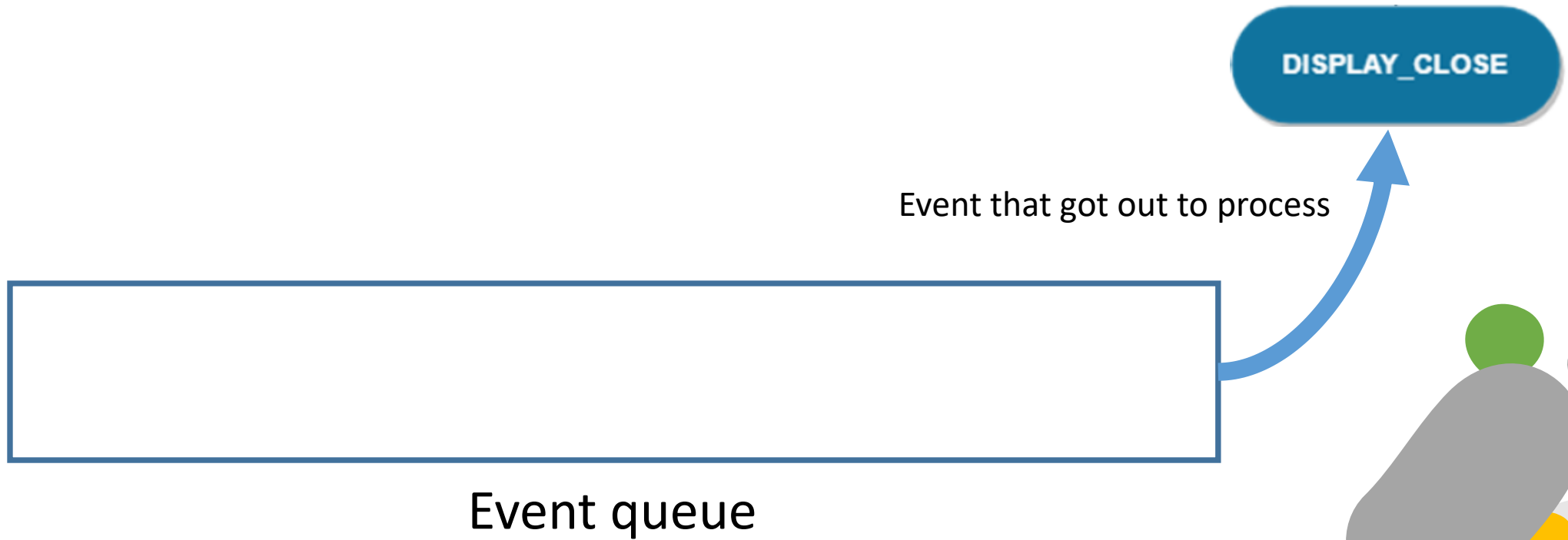
# Event queue

A queue that store event



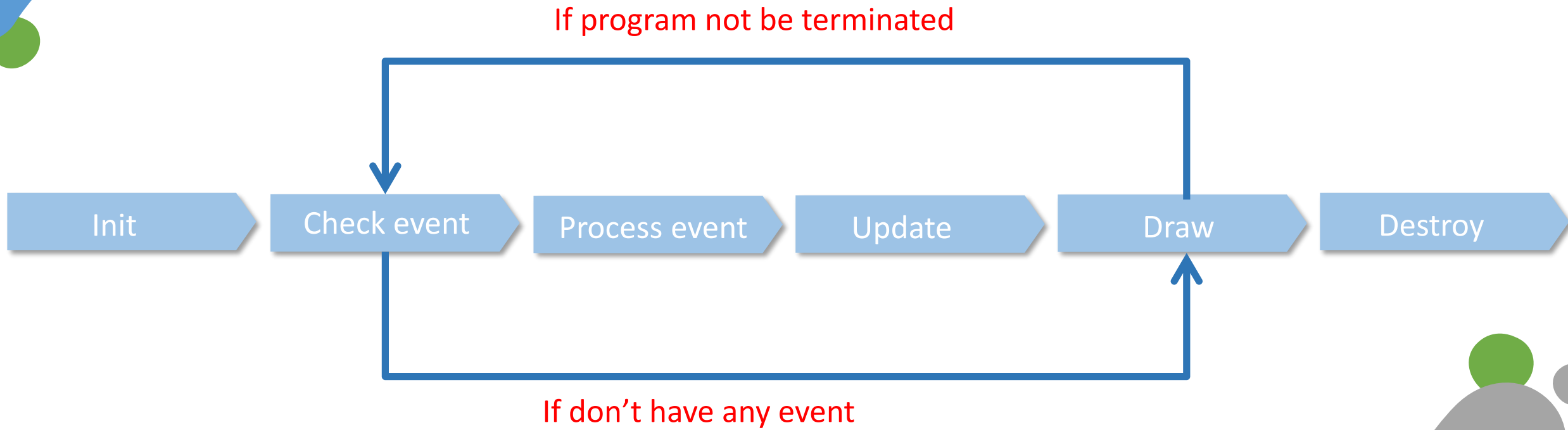
# Event queue

A queue that store event



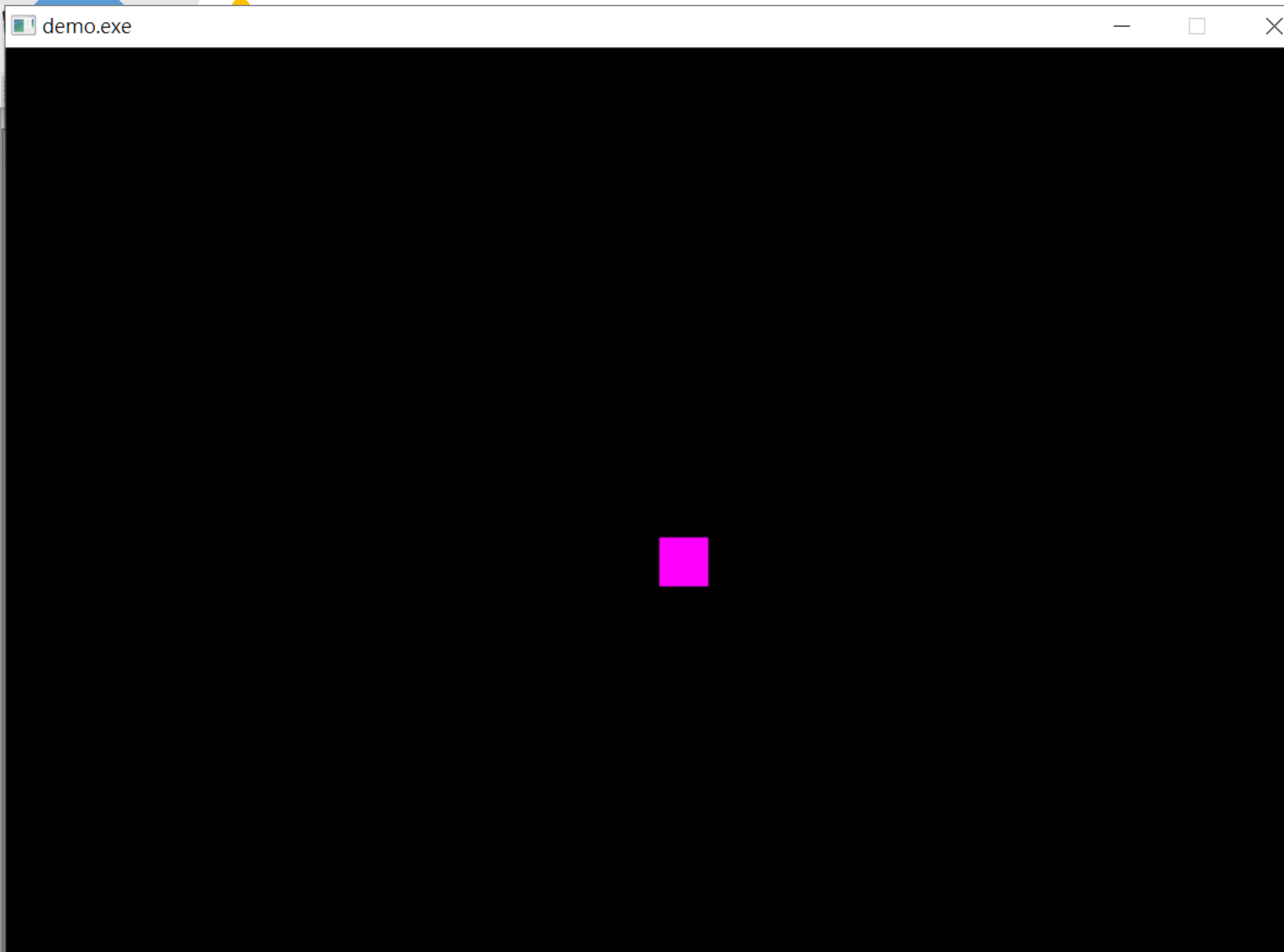
# event

The process flow of event





# event



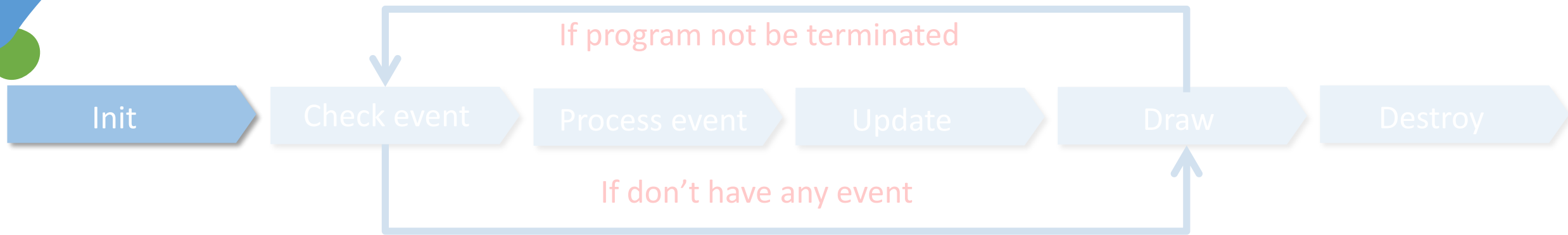
Take a program that you can move the rectangle around as example.

Use “WASD” to control the movement of rectangle.

Pressed “ESC” to terminate the program.

# event

The process flow of event

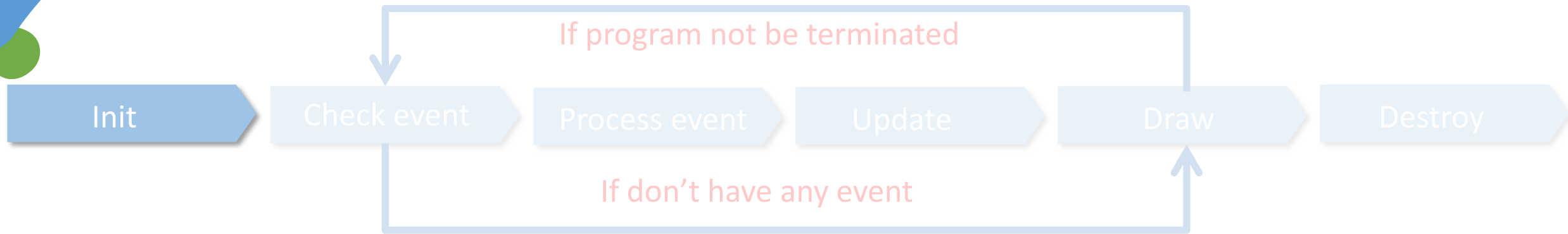


```
int main(int argc, char *argv[]) {  
    int msg = 0;  
    game_init();  
    while (msg != GAME_TERMINATE) {  
        msg = game_run();  
    }  
    game_destroy();  
    return 0;  
}
```

```
event_queue = al_create_event_queue();  
al_install_keyboard();  
al_register_event_source(event_queue, al_get_keyboard_event_source());
```

# event

The process flow of event



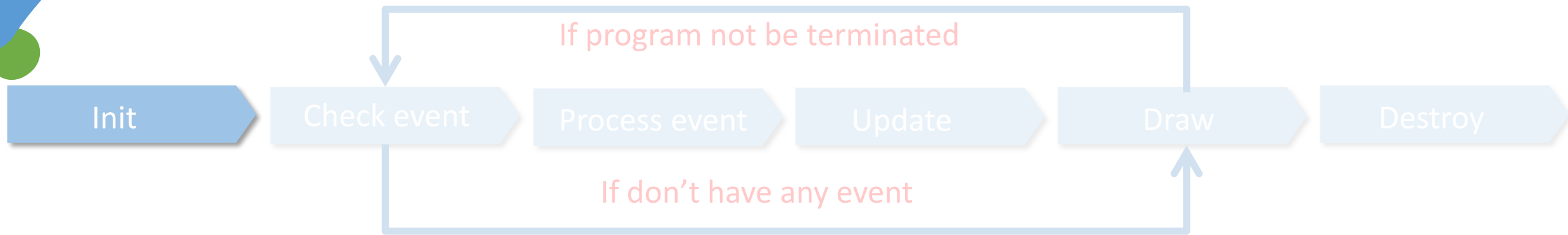
```
int main(int argc, char *argv[]) {  
    int msg = 0;  
    game_init();  
    while (msg != GAME_TERMINATE) {  
        msg = game_run();  
    }  
    game_destroy();  
    return 0;  
}
```

```
event_queue = al_create_event_queue();  
al_install_keyboard();  
al_register_event_source(event_queue, al_get_keyboard_event_source());
```

Create event queue

# event

The process flow of event



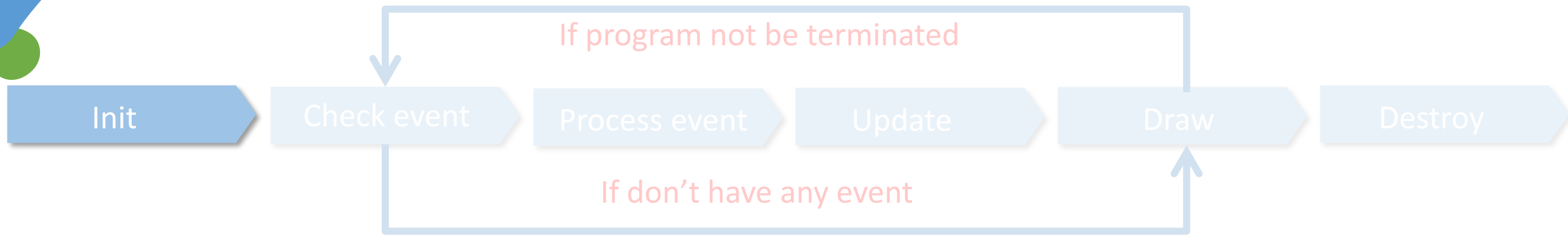
```
int main(int argc, char *argv[]) {  
    int msg = 0;  
    game_init();  
    while (msg != GAME_TERMINATE) {  
        msg = game_run();  
    }  
    game_destroy();  
    return 0;  
}
```

```
event_queue = al_create_event_queue();  
al_install_keyboard();  
al_register_event_source(event_queue, al_get_keyboard_event_source());
```

Install keyboard

# event

The process flow of event



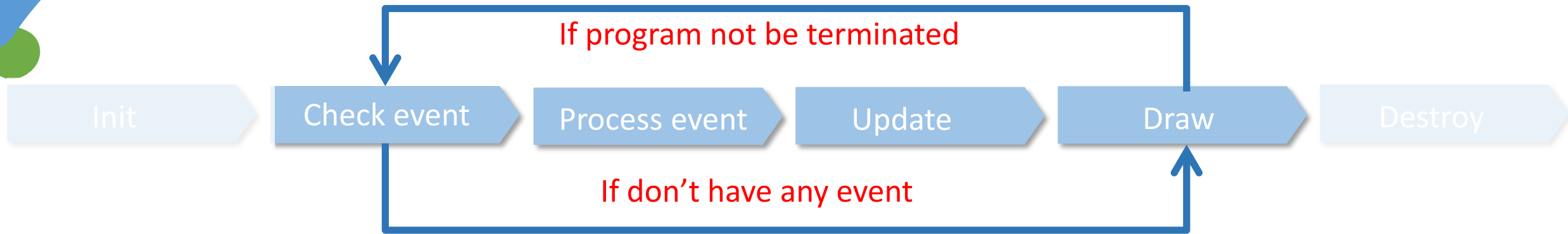
```
int main(int argc, char *argv[]) {  
    int msg = 0;  
    game_init();  
    while (msg != GAME_TERMINATE) {  
        msg = game_run();  
    }  
    game_destroy();  
    return 0;  
}
```

```
event_queue = al_create_event_queue();  
al_install_keyboard();  
al_register_event_source(event_queue, al_get_keyboard_event_source());
```

Register keyboard event by using  
`al_register_event_source()`,  
get the keyboard event source by using  
`al_get_keyboard_event_source()`

# event

The process flow of event

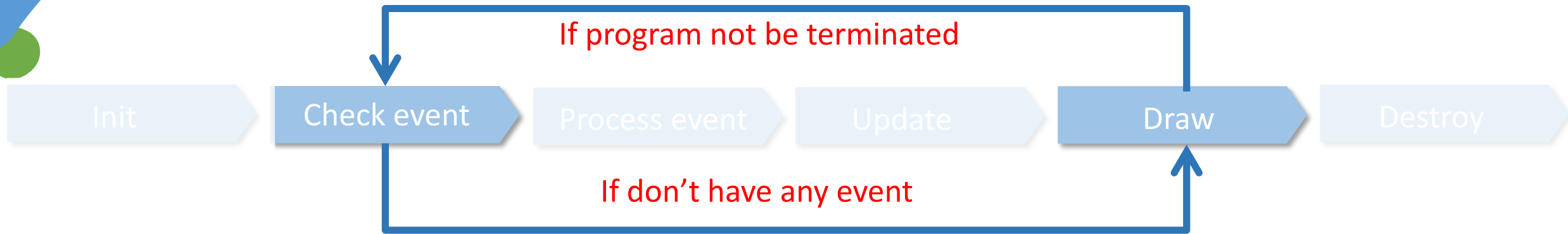


```
int main(int argc, char *argv[]) {  
    int msg = 0;  
    game_init();  
    while (msg != GAME_TERMINATE) {  
        msg = game_run();  
    }  
    game_destroy();  
    return 0;  
}
```

The loop that process the event

# event

The process flow of event



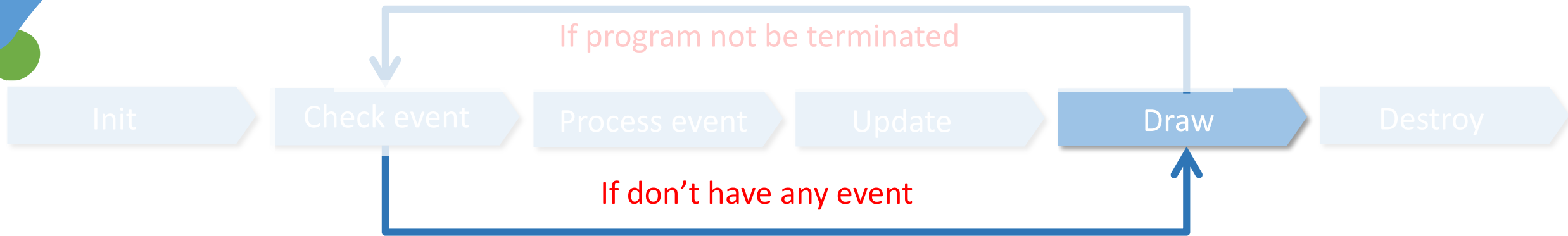
```
int game_run() {  
    int error = 0;  
    if (!al_is_event_queue_empty(event_queue)) {  
        error = process_event();  
    }  
    game_draw();  
    return error;  
}
```

Bool

`al_is_event_queue_empty(ALLEGRO_EVENT_QUEUE *)` is a bool function that check if there is any event in the event queue.

# event

The process flow of event



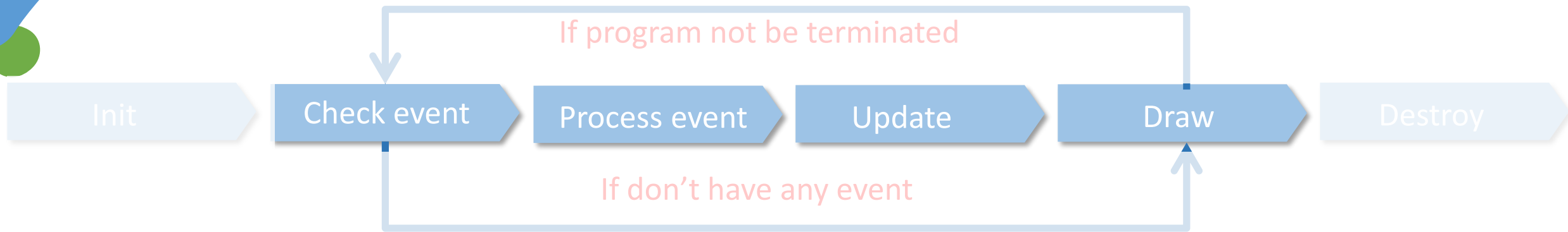
```
int game_run() {  
    int error = 0;  
    if (!al_is_event_queue_empty(event_queue)) {  
        error = process_event();  
    }  
    game_draw();  
    return error;  
}
```

If there doesn't exist any event then draw what you have



# event

The process flow of event

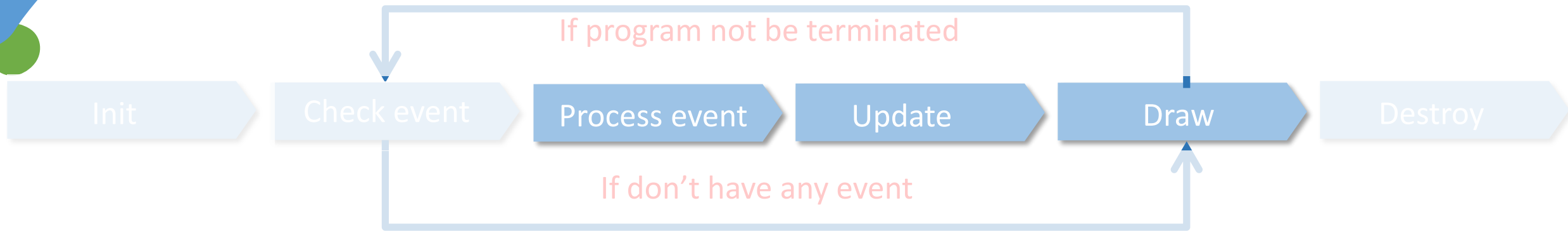


```
int game_run() {  
    int error = 0;  
    if (!al_is_event_queue_empty(event_queue)) {  
        error = process_event();  
    }  
    game_draw();  
    return error;  
}
```

If there' exists some event in the event queue, then process the event

# event

The process flow of event



```
int process_event() {  
    al_wait_for_event(event_queue, &event);  
    keyboard_event();  
    int error = game_update();  
    game_draw();  
    return error;  
}
```

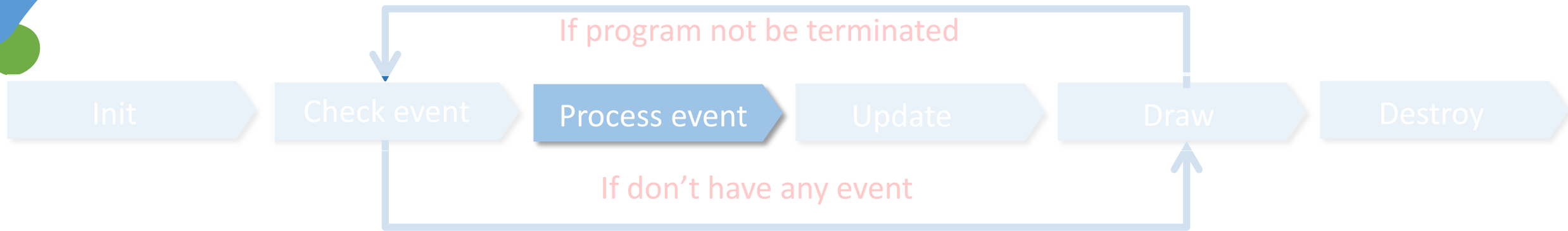
Use

`void al_wait_for_event(ALLEGRO_EVENT_QUEUE *queue, ALLEGRO_EVENT *ret_event)` to get the event

Your program will stop and wait for event.

# event

The process flow of event

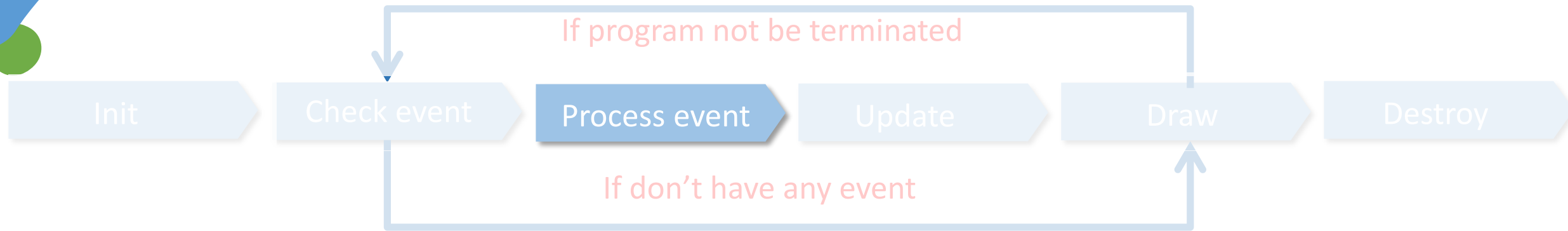


```
int process_event() {  
    al_wait_for_event(event_queue, &event);  
    keyboard_event();  
    int error = game_update();  
    game_draw();  
    return error;  
}
```

Process keyboard event

# event

The process flow of event

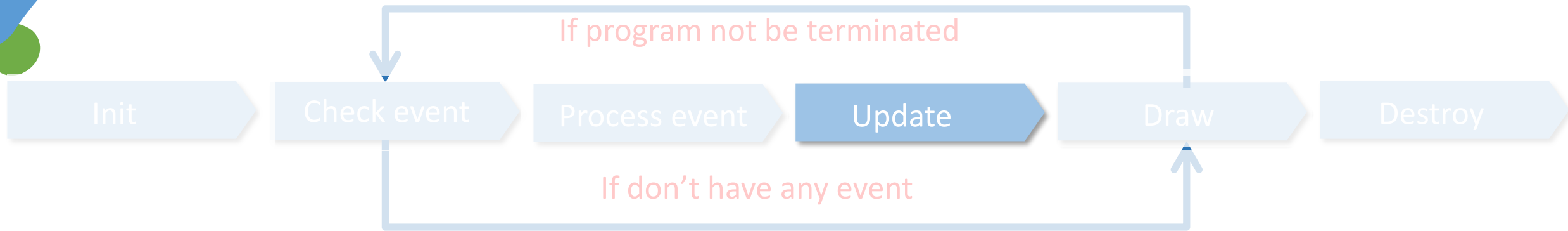


```
bool key_state[ALLEGRO_KEY_MAX];
int keyboard_event(){
    if( event.type == ALLEGRO_EVENT_KEY_DOWN ){
        key_state[event.keyboard.keycode] = true;
    }else if( event.type == ALLEGRO_EVENT_KEY_UP ){
        key_state[event.keyboard.keycode] = false;
    }
    return 0;
}
```

1. Declare a global array "key\_state" to determine the state of certain key. If the state is false means the key is being pressed, otherwise not being pressed
2. "event.type" can get the type of event
3. "event.keyboard.keycode" can get which key cause the event

# event

The process flow of event

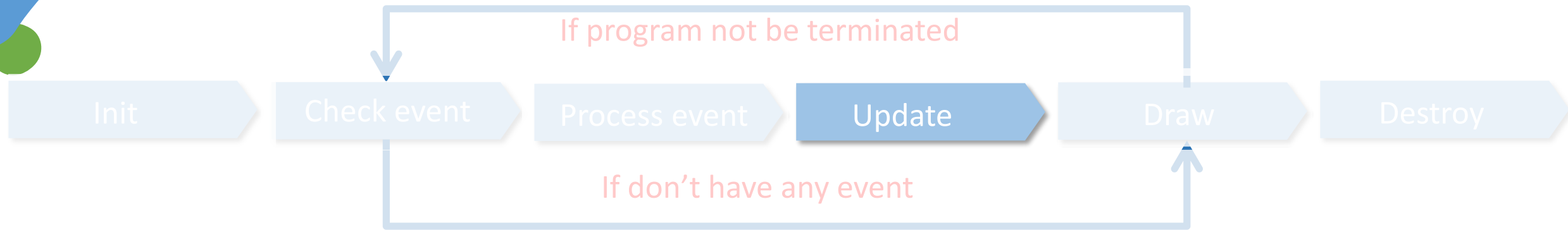


```
int process_event() {  
    al_wait_for_event(event_queue, &event);  
    keyboard_event();  
    int error = game_update();  
    game_draw();  
    return error;  
}
```

Update variables depend on the event

# event

The process flow of event

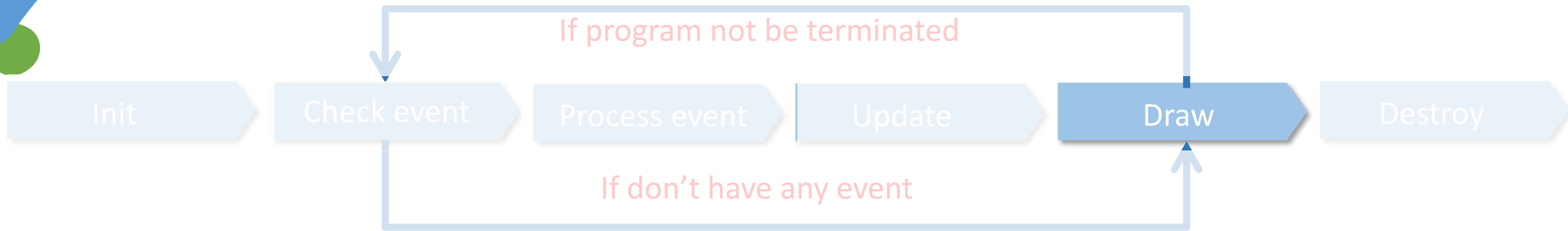


```
int game_update(){
    if( key_state[ALLEGRO_KEY_W] ){
        pos_y -= 10;
    }else if( key_state[ALLEGRO_KEY_A] ){
        pos_x -= 10;
    }else if( key_state[ALLEGRO_KEY_S] ){
        pos_y += 10;
    }else if( key_state[ALLEGRO_KEY_D] ){
        pos_x += 10;
    }else if( key_state[ALLEGRO_KEY_ESCAPE] ){
        return GAME_TERMINATE;
    }
    return 0;
}
```

Update the position of rectangle base on which key is being pressed.

# event

The process flow of event

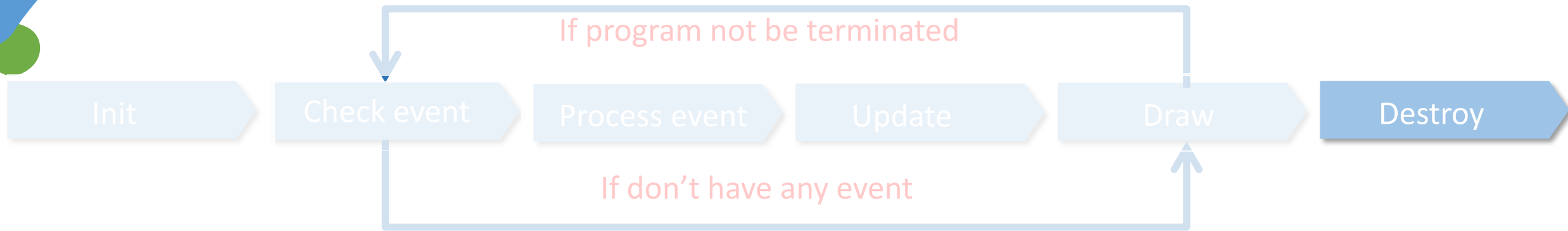


```
int process_event() {  
    al_wait_for_event(event_queue, &event);  
    keyboard_event();  
    int error = game_update();  
    game_draw();  
    return error;  
}
```

Draw what you want

# event

The process flow of event



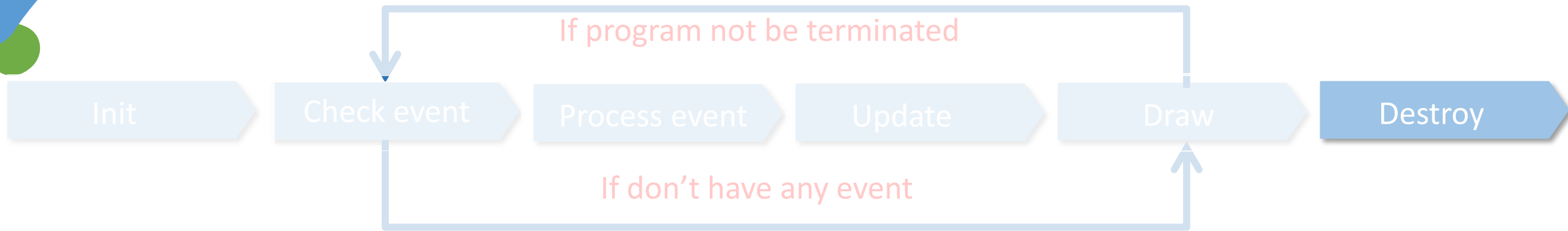
```
al_destroy_event_queue(event_queue);
```

→ Destroy event queue



# event

The process flow of event



```
al_destroy_event_queue(event_queue);
```

→ Destroy event queue



# Outline

01

Introduction

02

Draw something!

03

Deal with event !

04

Play some music!

05

Supplementary

# Display sound



# Display sound

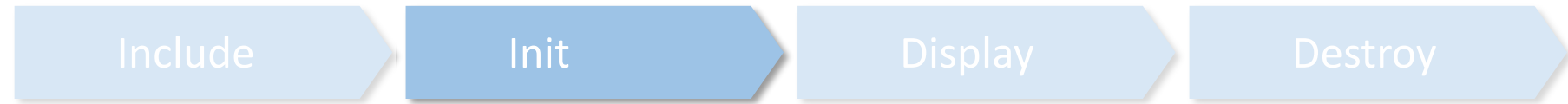


```
#include <allegro5/allegro_audio.h>  
#include <allegro5/allegro_acodec.h>
```



Include the head file can display sound

# Display sound



```
al_init_acodec_addon();
```



Initialize audio addon

# Display sound

Include

Init

Display

Destroy

```
sample = al_load_sample("growl.wav");  
startSound = al_create_sample_instance(sample);  
al_set_sample_instance_playmode(startSound, ALLEGRO_PLAYMODE_ONCE);  
al_attach_sample_instance_to_mixer(startSound, al_get_default_mixer());  
al_play_sample_instance(startSound);
```

Load the sound instance

# Display sound

Include

Init

Display

Destroy

```
sample = al_load_sample("growl.wav");  
startSound = al_create_sample_instance(sample);  
al_set_sample_instance_playmode(startSound, ALLEGRO_PLAYMODE_ONCE);  
al_attach_sample_instance_to_mixer(startSound, al_get_default_mixer());  
al_play_sample_instance(startSound);
```

Create a sound instance so that it  
can be attach into mixer

# Display sound

Include

Init

Display

Destroy

```
sample = al_load_sample("growl.wav");  
startSound = al_create_sample_instance(sample);  
al_set_sample_instance_playmode(startSound, ALLEGRO_PLAYMODE_ONCE);  
al_attach_sample_instance_to_mixer(startSound, al_get_default_mixer());  
al_play_sample_instance(startSound);
```

Set the play mode, you can use:  
ALLEGRO\_PLAYMODE\_ONCE  
ALLEGRO\_PLAYMODE\_LOOP



# Display sound

Include

Init

Display

Destroy

```
sample = al_load_sample("growl.wav");  
startSound = al_create_sample_instance(sample);  
al_set_sample_instance_playmode(startSound, ALLEGRO_PLAYMODE_ONCE);  
al_attach_sample_instance_to_mixer(startSound, al_get_default_mixer());  
al_play_sample_instance(startSound);
```

Attach the sound into mixer so that it  
can be played with multiple sound

# Display sound

Include

Init

Display

Destroy

```
sample = al_load_sample("growl.wav");  
startSound = al_create_sample_instance(sample);  
al_set_sample_instance_playmode(startSound, ALLEGRO_PLAYMODE_ONCE);  
al_attach_sample_instance_to_mixer(startSound, al_get_default_mixer());  
al_play_sample_instance(startSound);
```

Play the sample.

# Display sound



```
al_stop_sample_instance(startSound);
```



You can use this function to stop the sample

# Display sound

Include

Init

Display

Destroy

You can use this function to control the volume

```
al_set_sample_instance_gain(startSound, 1);
```

# Display sound



```
al_destroy_sample_instance(startSound);
```



Destroy the sample instance



# Outline

01

Introduction

02

Draw something!

03

Deal with event !

04

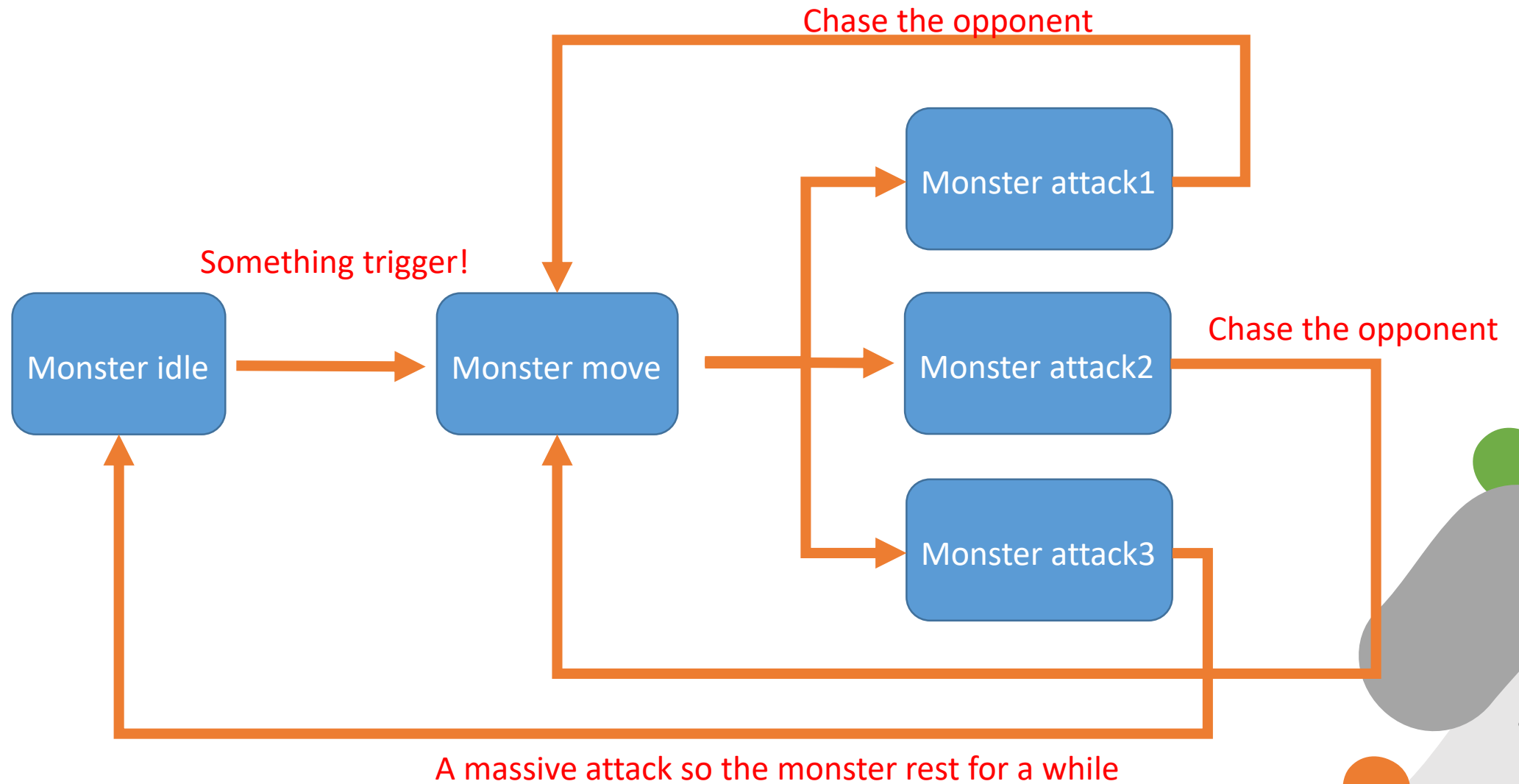
Play some music!

05

Supplementary

# AI

~Use finite state machine!!!



# AI

~Use finite state machine!!!

Monster idle

Draw the  
stop monster

Monster move

Draw the  
moving  
animation

Monster attack1

Draw the  
attack1  
animation

Monster attack2

Draw the  
attack2  
animation

Monster attack3

Draw the  
attack3  
animation

By using states, we can easy  
draw the animation of monster



# Special effect sound

~Use finite state machine!!!

Monster idle

Display the  
sound of  
panting

Monster move

Display the  
sound of  
moving

Monster attack1

Display the  
sound of  
attack1

Monster attack2

Display the  
sound of  
attack2

Monster attack3

Display the  
sound of  
attack3

Just judge the state then use  
`al_play_sample(*ALLEGRO_SAMPLE)`

# Display video

- Use something like mp4 to jpg to transfer the video into images.
- Load the images into an array. You can use `sprintf` to manipulate the path of image
- Set a timer as the fps of your video.
- Set a event queue to get the timer event.
- Each time the timer trigger display the image on the screen
- Plus the index of array by 1 to display the next image.
- Then you get the effect of display video!

# Display video

- Use allegro video addon!



# Display video

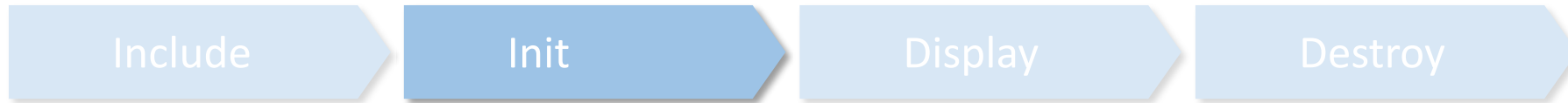
- Use allegro video addon!



```
#include <allegro5/allegro_audio.h>  
#include <allegro5/allegro_video.h>
```

# Display video

- Use allegro video addon!

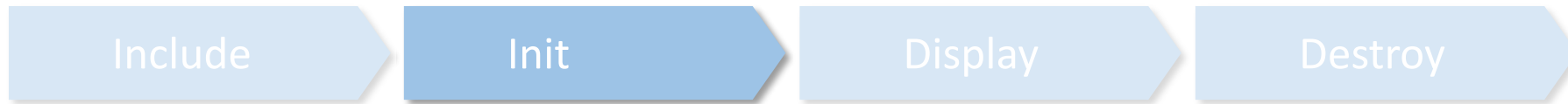


```
al_init_video_addon();  
ALLEGRO_VIDEO *video = al_open_video(filename);  
ALLEGRO_EVENT_SOURCE *temp = al_get_video_event_source(video);  
al_register_event_source(queue, temp);
```

Initialize video addon

# Display video

- Use allegro video addon!



```
al_init_video_addon();  
ALLEGRO_VIDEO *video = al_open_video(filename);  
ALLEGRO_EVENT_SOURCE *temp = al_get_video_event_source(video);  
al_register_event_source(queue, temp);
```



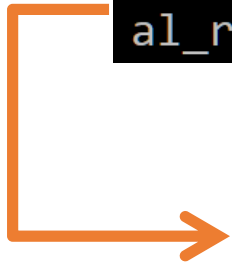
Open the video file  
(Note: only “.ogv” type of video is available)

# Display video

- Use allegro video addon!



```
al_init_video_addon();  
ALLEGRO_VIDEO *video = al_open_video(filename);  
ALLEGRO_EVENT_SOURCE *temp = al_get_video_event_source(video);  
al_register_event_source(queue, temp);
```



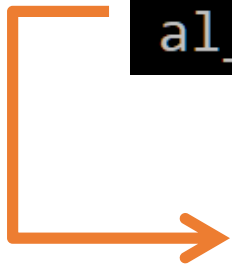
Get the event source of video and register it into event queue.

# Display video

- Use allegro video addon!



```
al_reserve_samples(1);  
al_start_video(video, al_get_default_mixer());
```



Reserve the sound channel for video and attach it into default mixer, then the video started



# Display video

- Use a loop to display the image of video



```
while(1){
    al_wait_for_event(queue, &event);
    if(event.type == ALLEGRO_EVENT_TIMER) {
        video_display(video);
    } else if( event.type == ALLEGRO_EVENT_DISPLAY_CLOSE ) {
        al_close_video(video);
        break;
    } else if( event.type == ALLEGRO_EVENT_VIDEO_FINISHED ) {
        break;
    }
}
```

→ The FPS of display the frame of video.

# Display video

- Use a loop to display the image of video



```
while(1){
    al_wait_for_event(queue, &event);
    if(event.type == ALLEGRO_EVENT_TIMER) {
        video_display(video);
    } else if( event.type == ALLEGRO_EVENT_DISPLAY_CLOSE ) {
        al_close_video(video);
        break;
    } else if( event.type == ALLEGRO_EVENT_VIDEO_FINISHED ) {
        break;
    }
}
```

If the display be closed, then close the video and break the loop

# Display video

- Use a loop to display the image of video



```
while(1){
    al_wait_for_event(queue, &event);
    if(event.type == ALLEGRO_EVENT_TIMER) {
        video_display(video);
    } else if( event.type == ALLEGRO_EVENT_DISPLAY_CLOSE ) {
        al_close_video(video);
        break;
    } else if( event.type == ALLEGRO_EVENT_VIDEO_FINISHED ) {
        break;
    }
}
```



If the video finished then break the loop

# Display video

- Use a loop to display the image of video



```
void video_display(ALLEGRO_VIDEO *video) {
    ALLEGRO_BITMAP *frame = al_get_video_frame(video);
    if (!frame)
        return;
    al_draw_scaled_bitmap(frame,
                          0, 0,
                          al_get_bitmap_width(frame),
                          al_get_bitmap_height(frame),
                          0, 0,
                          al_get_display_width(screen),
                          al_get_display_height(screen), 0);
    al_flip_display();
}
```

Get one frame from the video, frame may be none therefore you should determine the value before draw

# Display video

- Use a loop to display the image of video



```
void video_display(ALLEGRO_VIDEO *video) {
    ALLEGRO_BITMAP *frame = al_get_video_frame(video);
    if (!frame)
        return;
    al_draw_scaled_bitmap(frame,
                          0, 0,
                          al_get_bitmap_width(frame),
                          al_get_bitmap_height(frame),
                          0, 0,
                          al_get_display_width(screen),
                          al_get_display_height(screen), 0);
    al_flip_display();
}
```

The position of original bitmap you want to scale

# Display video

- Use a loop to display the image of video



```
void video_display(ALLEGRO_VIDEO *video) {
    ALLEGRO_BITMAP *frame = al_get_video_frame(video);
    if (!frame)
        return;
    al_draw_scaled_bitmap(frame,
                          0, 0,
                          al_get_bitmap_width(frame),
                          al_get_bitmap_height(frame),
                          0, 0,
                          al_get_display_width(screen),
                          al_get_display_height(screen), 0);
    al_flip_display();
}
```

The width/height of original bitmap you want to scale

# Display video

- Use a loop to display the image of video



```
void video_display(ALLEGRO_VIDEO *video) {
    ALLEGRO_BITMAP *frame = al_get_video_frame(video);
    if (!frame)
        return;
    al_draw_scaled_bitmap(frame,
                          0, 0,
                          al_get_bitmap_width(frame),
                          al_get_bitmap_height(frame),
                          0, 0,
                          al_get_display_width(screen),
                          al_get_display_height(screen), 0);
    al_flip_display();
}
```

The position of the result image on the display



# Display video

- Use a loop to display the image of video



```
void video_display(ALLEGRO_VIDEO *video) {
    ALLEGRO_BITMAP *frame = al_get_video_frame(video);
    if (!frame)
        return;
    al_draw_scaled_bitmap(frame,
                          0, 0,
                          al_get_bitmap_width(frame),
                          al_get_bitmap_height(frame),
                          0, 0,
                          al_get_display_width(screen),
                          al_get_display_height(screen), 0);
    al_flip_display();
}
```

The width/height of the result image on the display



# Display video

- Use a loop to display the image of video



```
void video_display(ALLEGRO_VIDEO *video) {
    ALLEGRO_BITMAP *frame = al_get_video_frame(video);
    if (!frame)
        return;
    al_draw_scaled_bitmap(frame,
                          0, 0,
                          al_get_bitmap_width(frame),
                          al_get_bitmap_height(frame),
                          0, 0,
                          al_get_display_width(screen),
                          al_get_display_height(screen), 0);
    al_flip_display();
}
```

The same flag of draw bitmap, control the flip

# Display video

Include

Init

Display

Destroy

```
al_draw_scaled_bitmap(frame,  
0, 0,  
al_get_bitmap_width(frame),  
al_get_bitmap_height(frame),  
0, 0,  
al_get_display_width(screen),  
al_get_display_height(screen), 0);
```

testing.exe



```
al_draw_scaled_bitmap(frame,  
100, 0,  
al_get_bitmap_width(frame)-100,  
al_get_bitmap_height(frame),  
0, 0,  
al_get_display_width(screen),  
al_get_display_height(screen), 0);
```

testing.exe



# Display video



- Close the video will automatically destroy the video.



The End~