

Event-driven finite state machine

Event-driven finite state machine



Event-driven finite state machine

Mit kell tudni ezen lecke megértéséhez?

Event-driven finite state machine

Mit kell tudni ezen lecke megértéséhez:

- program olvasás (C, Python, Java, JS stb.)

Event-driven finite state machine

Mit kell tudni ezen lecke megértéséhez:

- program olvasás (C, Python, Java, JS stb.)
- variable, condition, branch, loop, function

Event-driven finite state machine

Mit kell tudni ezen lecke megértéséhez:

- program olvasás (C, Python, Java, JS stb.)
- variable, condition, branch, loop, function



Event-driven finite state machine

- event alapok
- event technikák
- finite state machine
- event-driven finite state machine

Event-driven finite state machine

- **event alapok**
- event technikák
- finite state machine
- event-driven finite state machine

Mi az event?

Mi az event?



Mi az event?



<spoiler>
 embedded megvalósítás: int + enum
</spoiler>

Mi az event?

Külső eseményt reprezentál, tartalma:

- **típus** (véges, előre definiált)
- **paraméterek** (típusonként, opcionális)
- **metaadatok** (framework kezeli, rejtett)

Event példa

Típus:
system shutdown

Paraméter:

-

Event példa

Típus:
mouse click

Paraméter:
x
y

Event példa

Típus:
mouse doubleclick

Paraméter:
x
y

Event példa

Típus:

mouse click combo

Paraméter:

click type (single/double)

x

y

Az event útja



Az event útja

- emitter: külvilág rezdüléseiiből eventet fabrikál

Az event útja

- emitter: külvilág rezdüléseiiből eventet fabrikál
- queue: gyűjtés-tárolás (FIFO)

Az event útja

- emitter: külvilág rezdüléseiiből eventet fabrikál
- queue: gyűjtés-tárolás (FIFO)
- dispatcher: eljuttatja a handlerhez

Az event útja

- emitter: külvilág rezdüléseiiből eventet fabrikál
- queue: gyűjtés-tárolás (FIFO)
- dispatcher: eljuttatja a handlerhez
- handler: feldolgozza

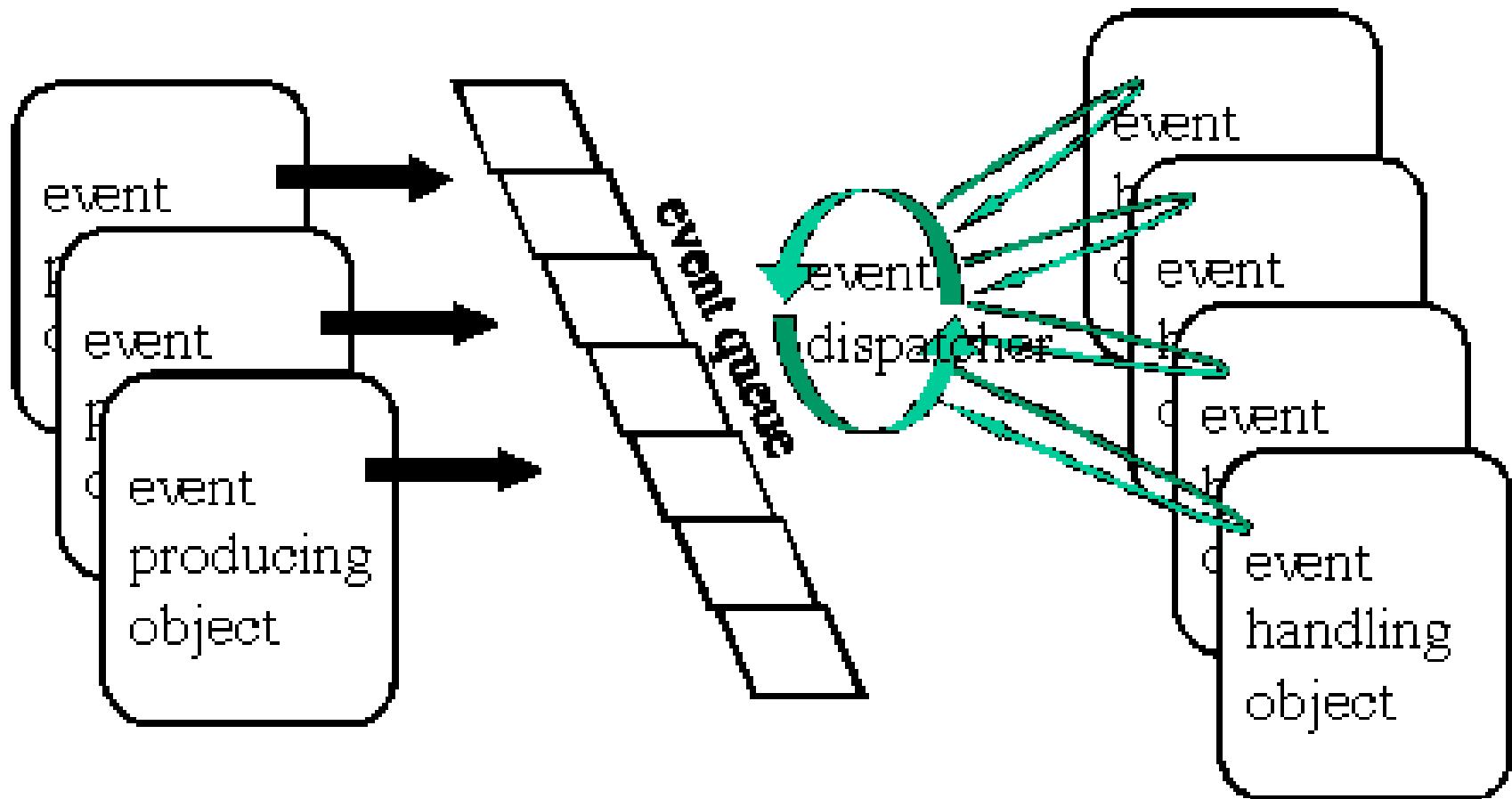
Framework

Framework event management funkciók:

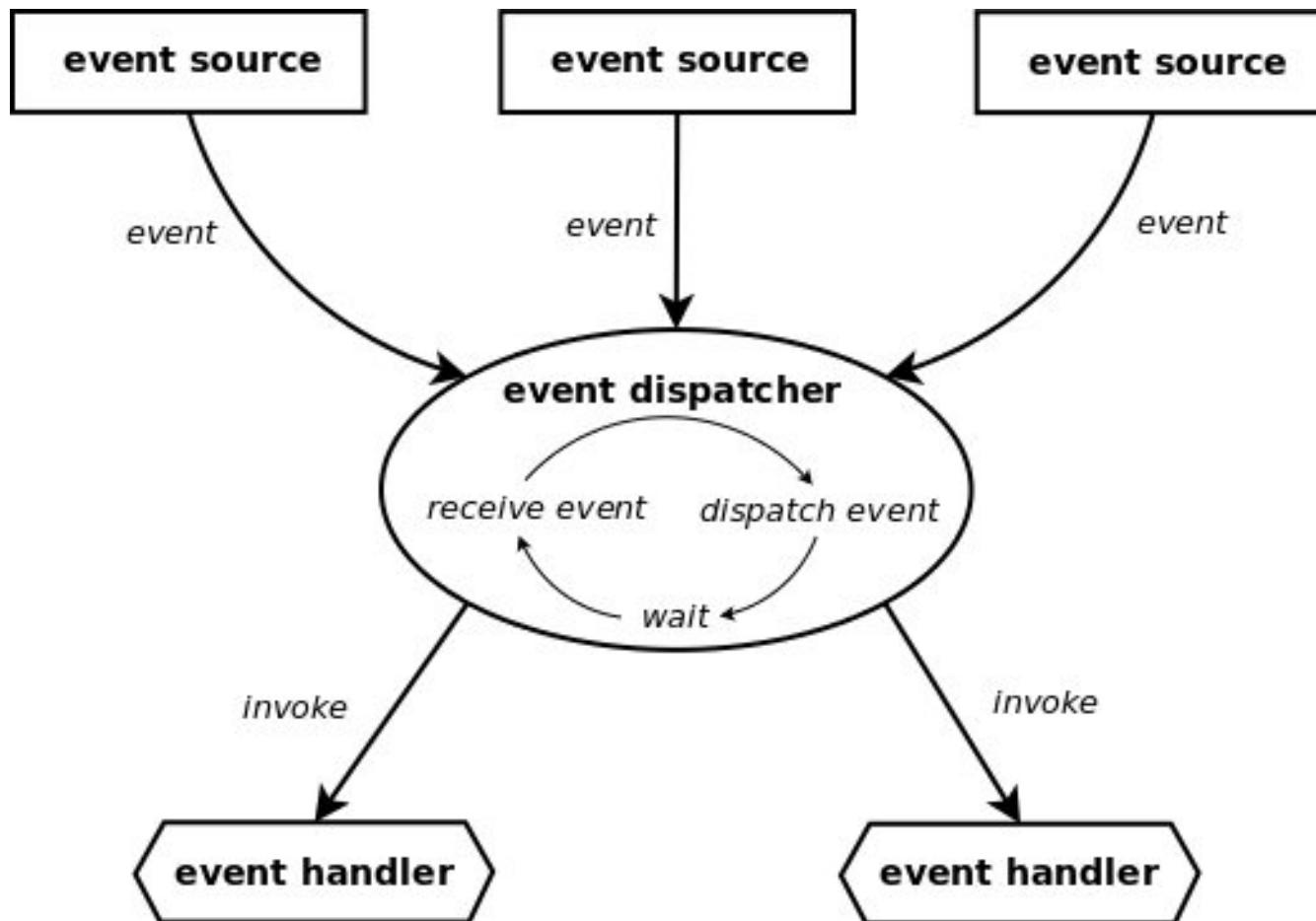
- előre definiált eventek (UI komponensekkel)
- event factory
- dispatcher: handler subscribe-ol event típusra
- default handler
- loggolás, debuggolás

Embedded platformokon: saját event rendszer

Az event útja



Az event útja



Az event útja: emitter

A külvilág történéseiiből:

- sensor
- timer
- device (serial, keyboard, touchscreen)
- network (request)

eventet készít, átadja a rendszernek, ennyike.



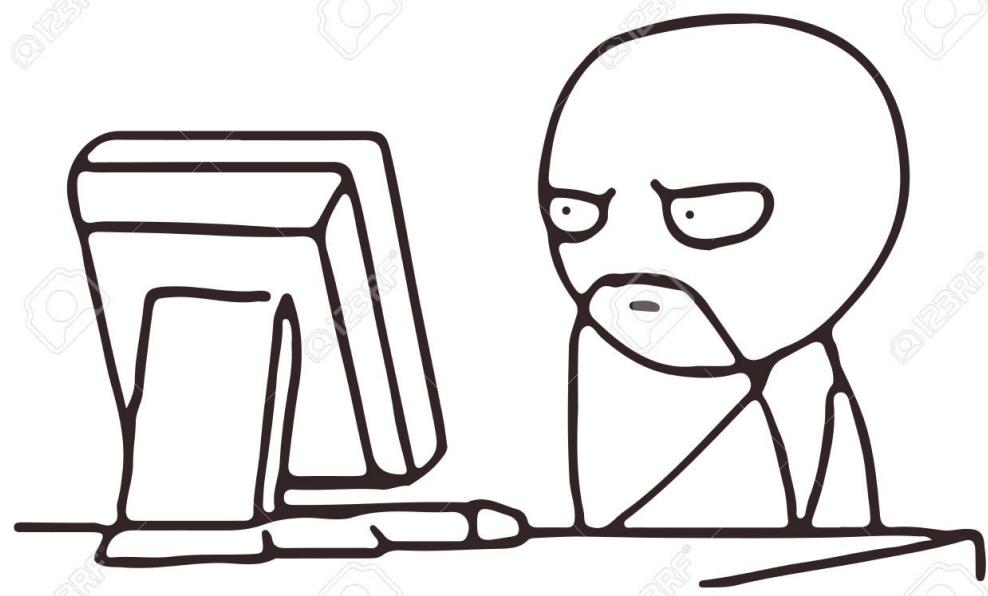
Az event útja: emitter

- állapot alapján változás megállapítása
- fals észlelések kiszűrése (prellegő pedál)
- magasabb szint kezelése (press-release vs click)
- timeout, hibakezelés

Event reprezentáció

- OOP: MouseEvent extends Event
event.getType(), mouseEvent.getX()
- embedded: int + enum (vagy amit kitalálsz)

```
enum EventType {  
    E_NONE,  
    E_IDLE,  
    E_CLICK,  
    E_HOLD,  
    E_GAMESTART  
};
```



Az event útja: dispatcher

- kiveszi a queue-ból
- átadja a handlernek

```
while (true) {
    // emitterek hívása

    switch (event) {
        case E_CLICK:
            procClick();
            break;
        case E_HOLD:
            procHold();
            break;
        case E_GAMESTART:
            procGameStart();
            break;
    }
    event = E_NONE;
}
```

Framework: event loop

- subscribe:
event type - handler
összerendelés
- a dispatcher a
rendszer része
- beavatkozási pont

```
while (true) {  
    e = sys.getNextEvent()  
    if (e == null) continue;  
    sys.dispatchEvent(e);  
}
```

Framework: event loop

- subscribe:
event type - handler
összerendelés
- a dispatcher a
rendszer része
- beavatkozási pont

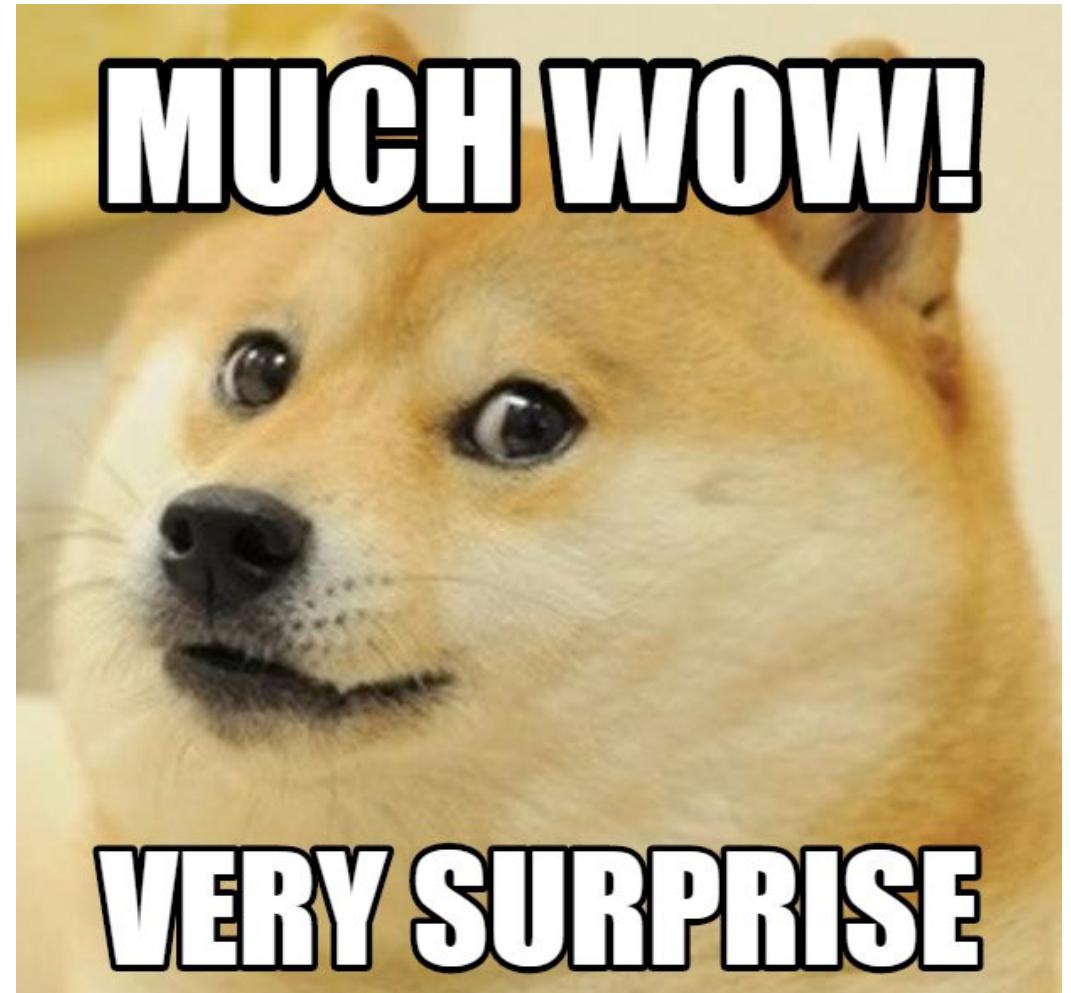
```
while (true) {  
    e = sys.getNextEvent()  
    if (e == null) continue;  
    if (e.type == E_QUIT) {  
        sys.log("quitting");  
        exit(0);  
    }  
    sys.dispatchEvent(e);  
}
```

Az event útja: handler

- feldolgozza

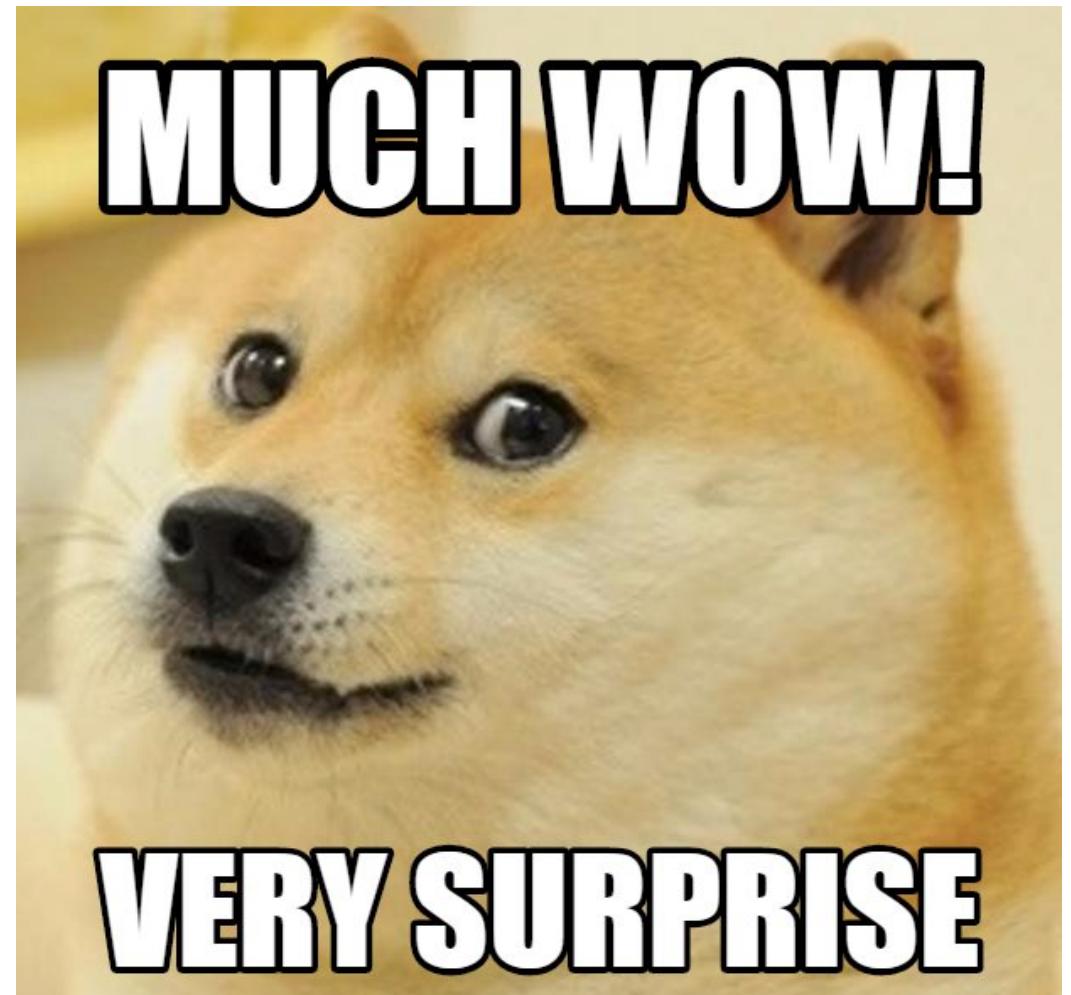
Az event útja: handler

- feldolgozza

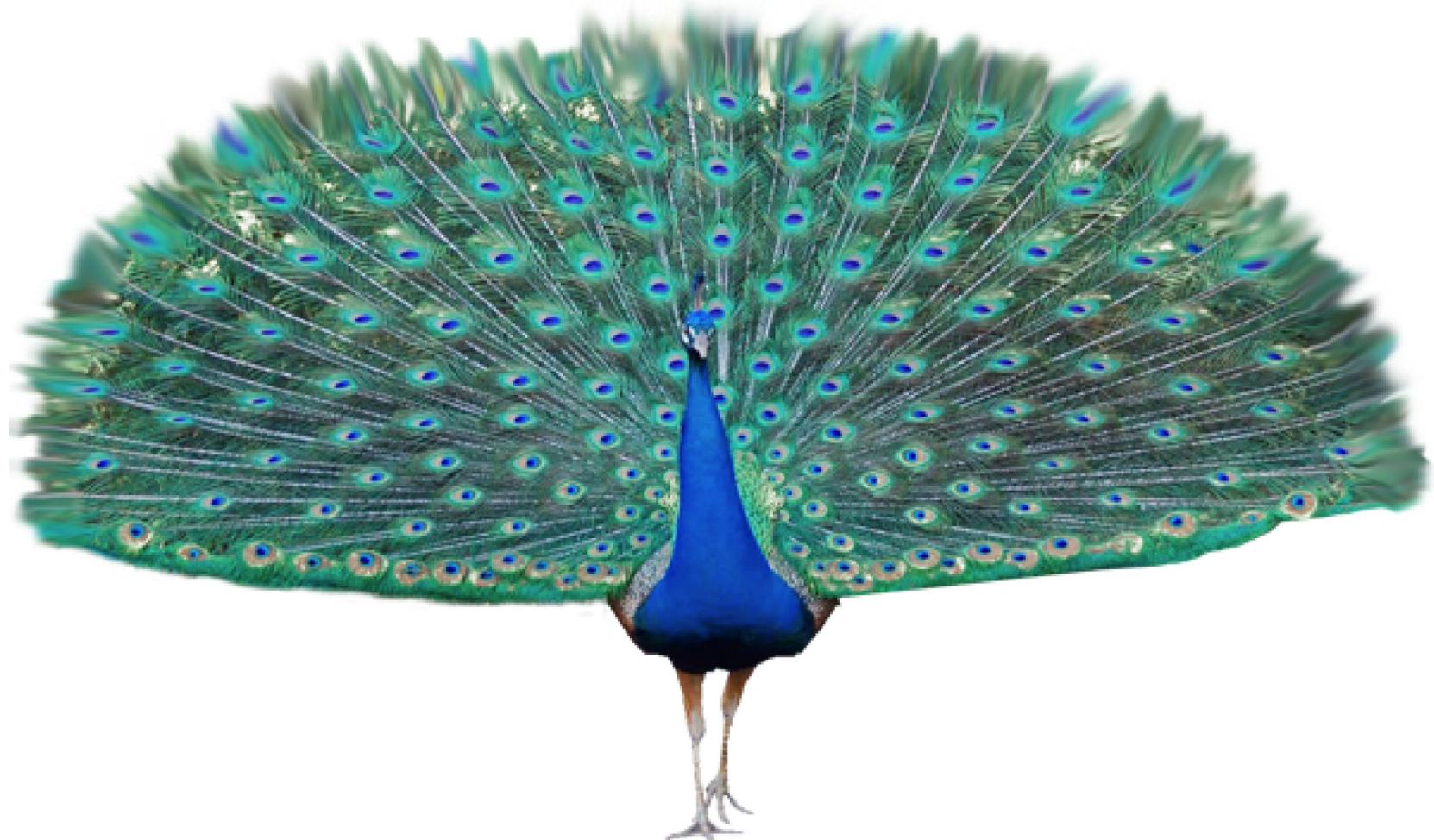


Az event útja: handler

- feldolgozza
- gyorsan



Mire jó ez az egész?



Mire jó ez az egész?

hagyományos

```
key = getkey();

if (key == 'z') {
    x--;
}

if (key == 'x') {
    x++;
}
```

event

```
key = getkey();
if (k == 'z') event = E_LEFT;
if (k == 'x') event = E_RIGHT;

switch (event) {
    case E_LEFT:
        x--;
        break;
    case E_RIGHT:
        x++;
        break;
}
```

Mire jó ez az egész?

```
event = E_NONE;

if (keypressed()) {
    key = getkey();
    if (key == 'z') event = E_LEFT;
    if (key == 'x') event = E_RIGHT;
}
```

```
switch (event) {
    case E_LEFT:
        x--;
        break;
    case E_RIGHT:
        x++;
        break;
}
```

Mire jó ez az egész?

emitters

```
event = E_NONE;

if (keypressed()) {
    key = getkey();
    if (key == 'z') event = E_LEFT;
    if (key == 'x') event = E_RIGHT;
}
```

handlers

```
switch (event) {
    case E_LEFT:
        x--;
        break;
    case E_RIGHT:
        x++;
        break;
}
```

Mire jó ez az egész?

emitters

```
event = E_NONE;

if (keypressed()) {
    key = getkey();
    if (key == 'z') event = E_LEFT;
    if (key == 'x') event = E_RIGHT;
}

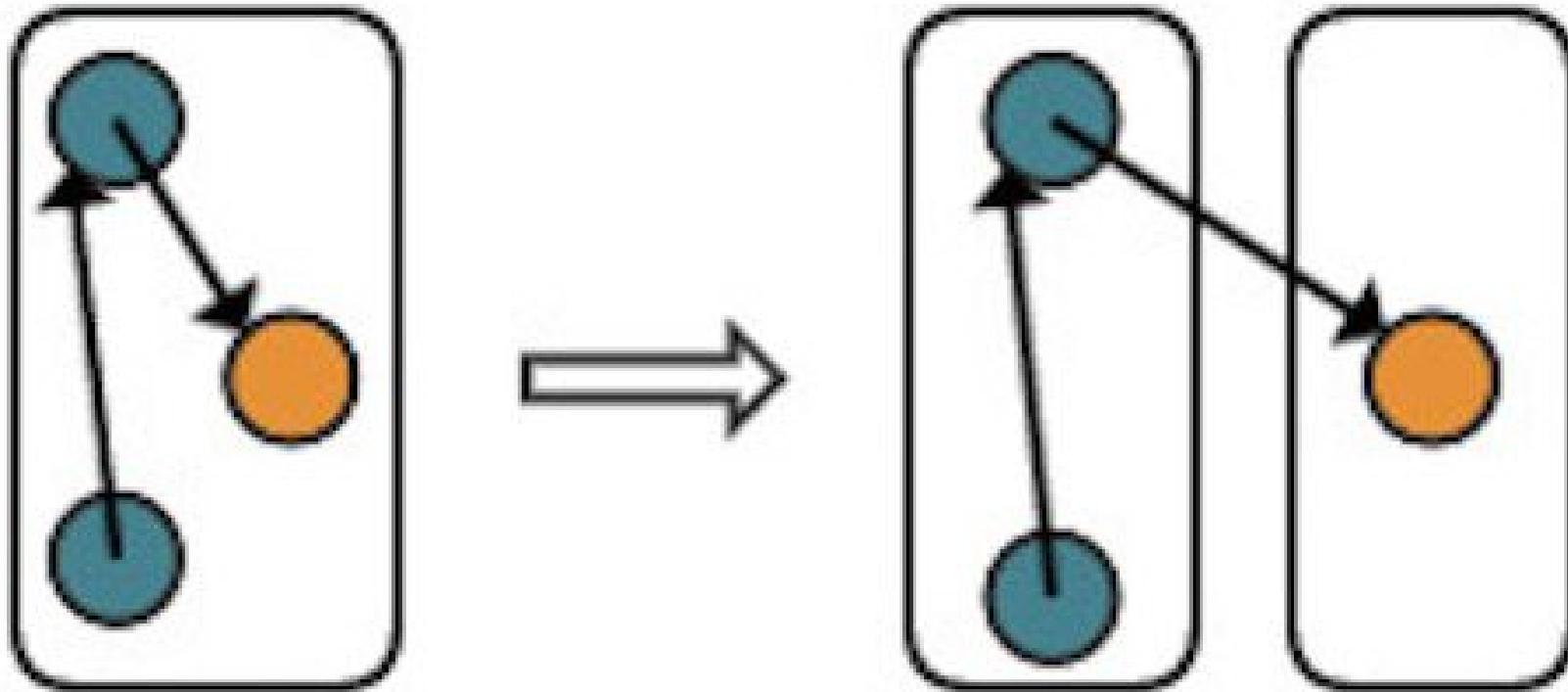
joy = getJoystickState();
if (joy == 1) event = E_LEFT;
if (joy == 3) event = E_RIGHT;
```

handlers

```
switch (event) {
    case E_LEFT:
        x--;
        break;
    case E_RIGHT:
        x++;
        break;
}
```

Mi ez az egész?

Loose coupling



Loose coupling

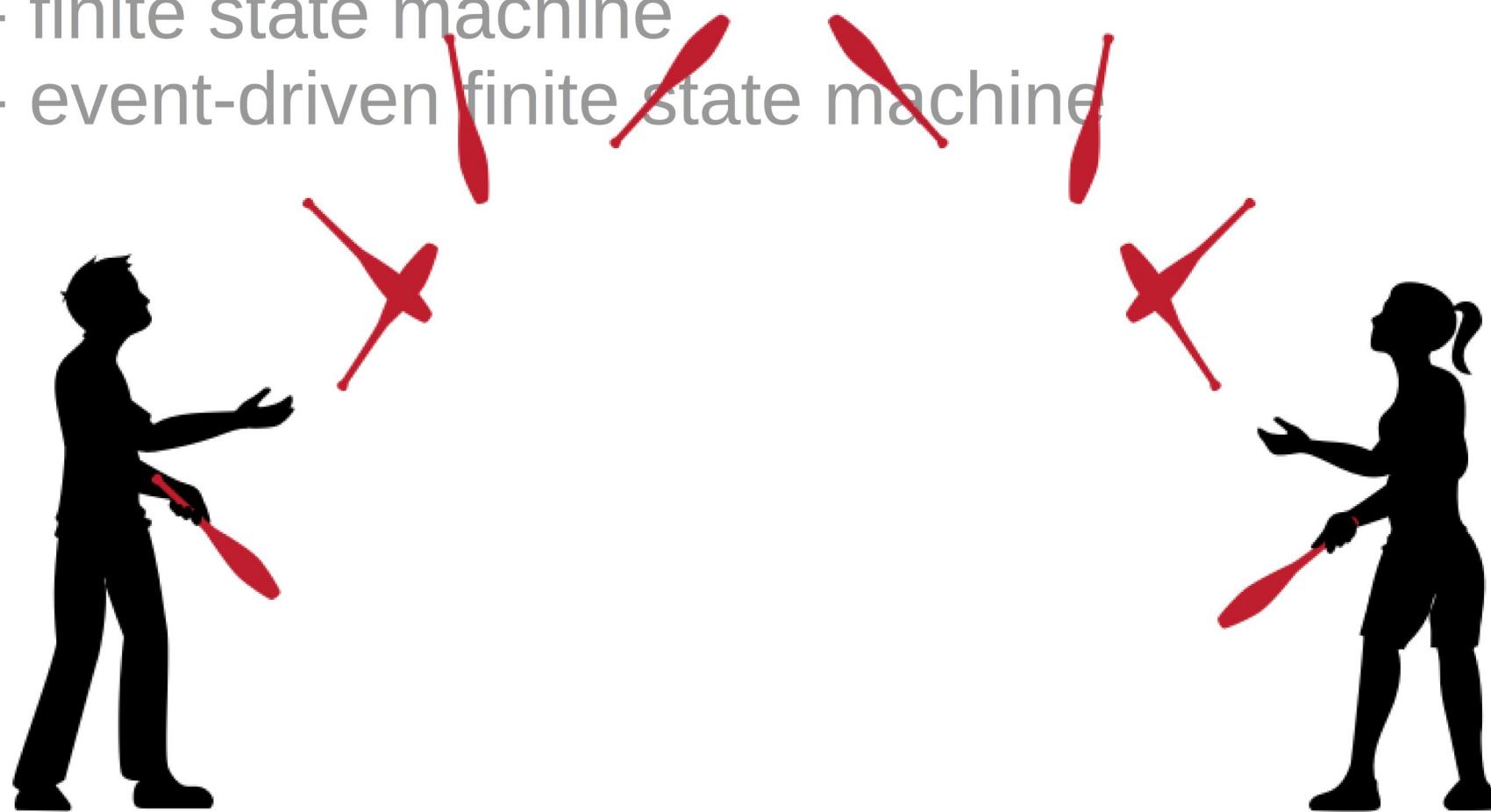


Mire jó ez az egész?

- Láthatóvá válik a program hatásköre:
az eventek összessége
- a program működése magas szinten követhető:
eventek sorozata
- a program belső szerkezete rendszerezett,
könnyen bővíthető, mindennek van saját neve

Event-driven finite state machine

- event alapok
- event technikák**
- finite state machine
- event-driven finite state machine



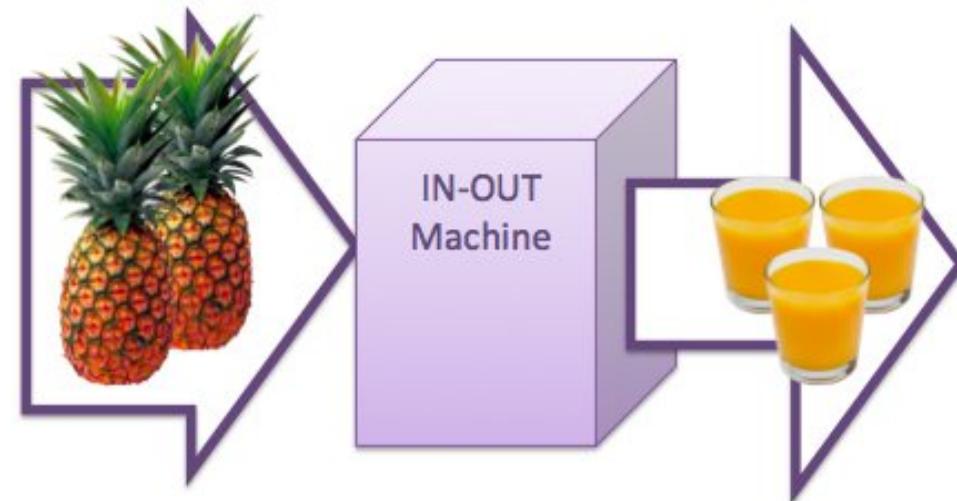
Event technikák

- filter: elnyelés bizonyos körülmények esetén



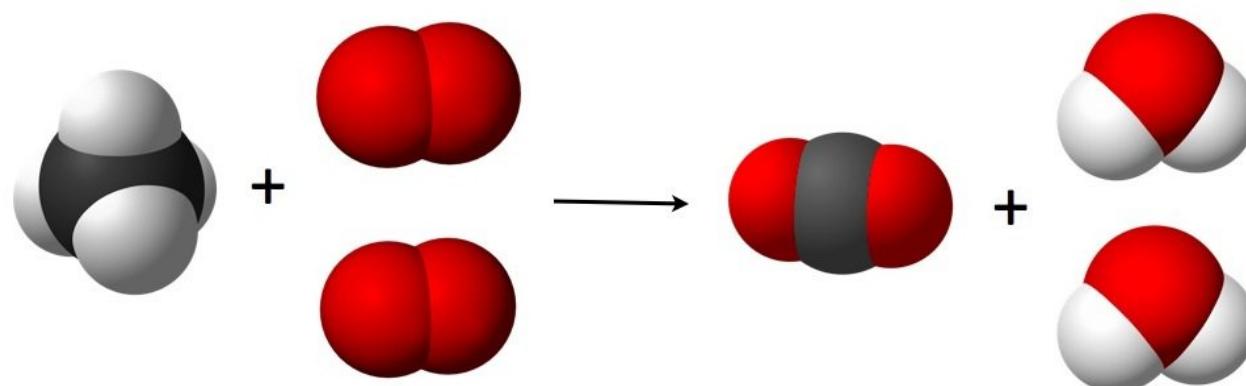
Event technikák

- filter: elnyelés bizonyos körülmények esetén
- transform:
 - eredeti event átalakítása, infó hozzáadása



Event technikák

- filter: elnyelés bizonyos körülmények esetén
- transform:
 - eredeti event átalakítása, infó hozzáadása
 - eventek alapján új event készítése
(mouse up + down => click, doubleclick)



Event technikák

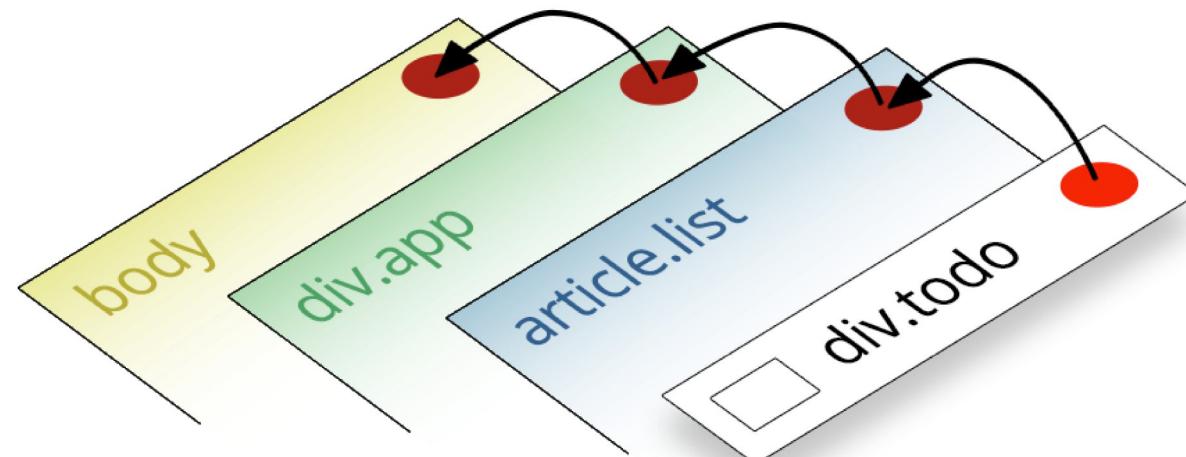
- filter: elnyelés bizonyos körülmények esetén
- transform:
 - eredeti event átalakítása, infó hozzáadása
 - eventek alapján új event készítése
(mouse up + down => click, doubleclick)
- redirect: más rendszerekbe irányítás



Bubbling

- lekezeljük és nyugtázzuk
- lekezeljük, de nem nyugtázzuk: továbbengedés
- nem kezeljük
- nem kezeljük, de nyugtázzuk: tiltás

browser: e.preventDefault(), e.stopPropagation()



ANR

Application Not Responding: event loop beáll

```
while (true) {  
    e = sys.getNextEvent()  
    if (e == null) continue;  
    sys.dispatchEvent(e);  
}
```

Example isn't
responding.

Do you want to close it?

Wait

OK

Message types

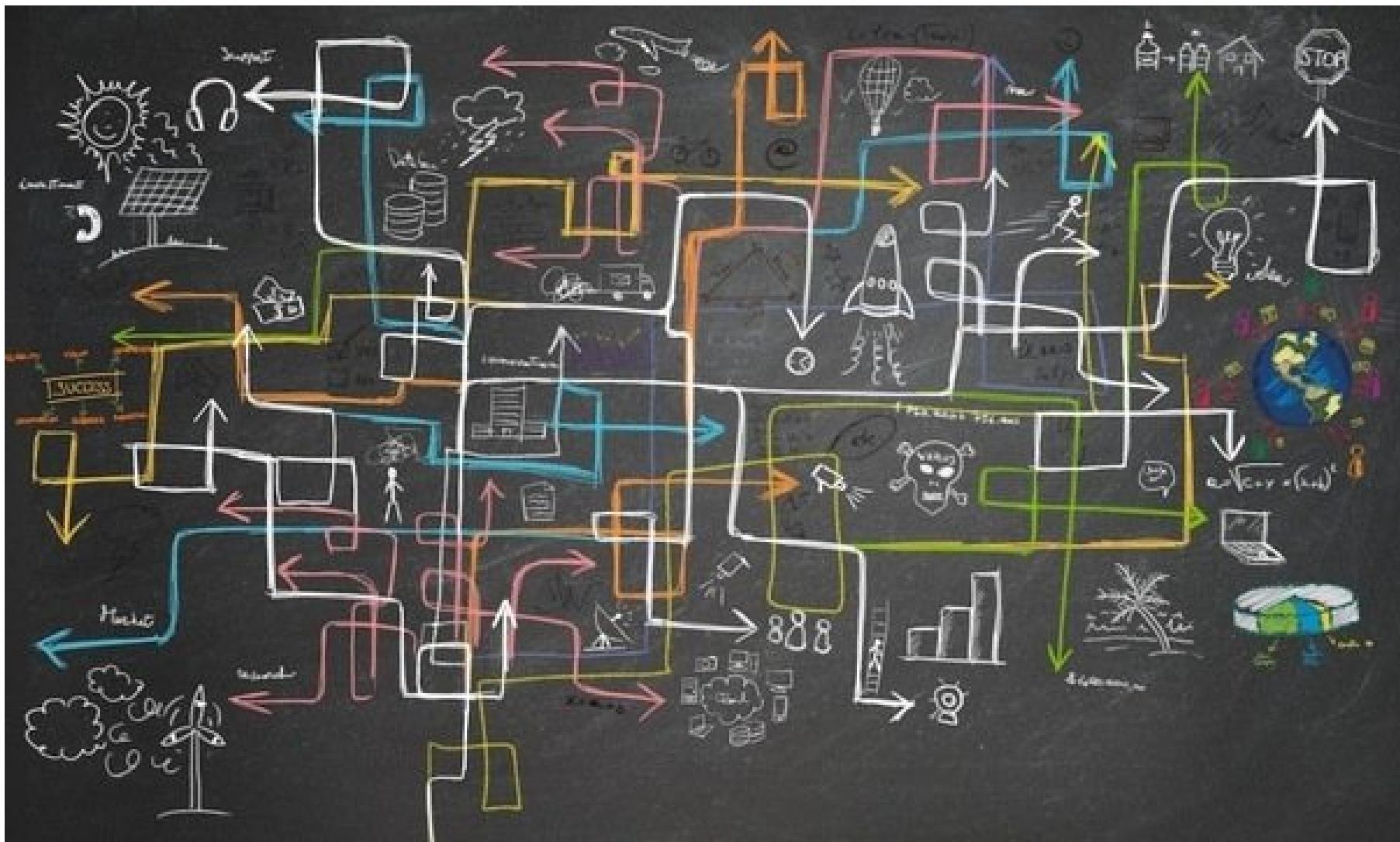
- **Message**: absztrakt, valaki küldi, valaki fogadja
- **Event**: mi történt (valaki majd lekezeli)
- **Command**: mit kell csinálni (valaki kiadta)
- Request-response: általában kívülről
- Signal: Unix process control
- Szimpla method call: küldünk egy message-et az objektumnak
- RPC

Event-driven finite state machine

- event alapok
- virtuóz event technikák
- finite state machine**
- event-driven finite state machine

Mi a state machine?

Mi a state machine?



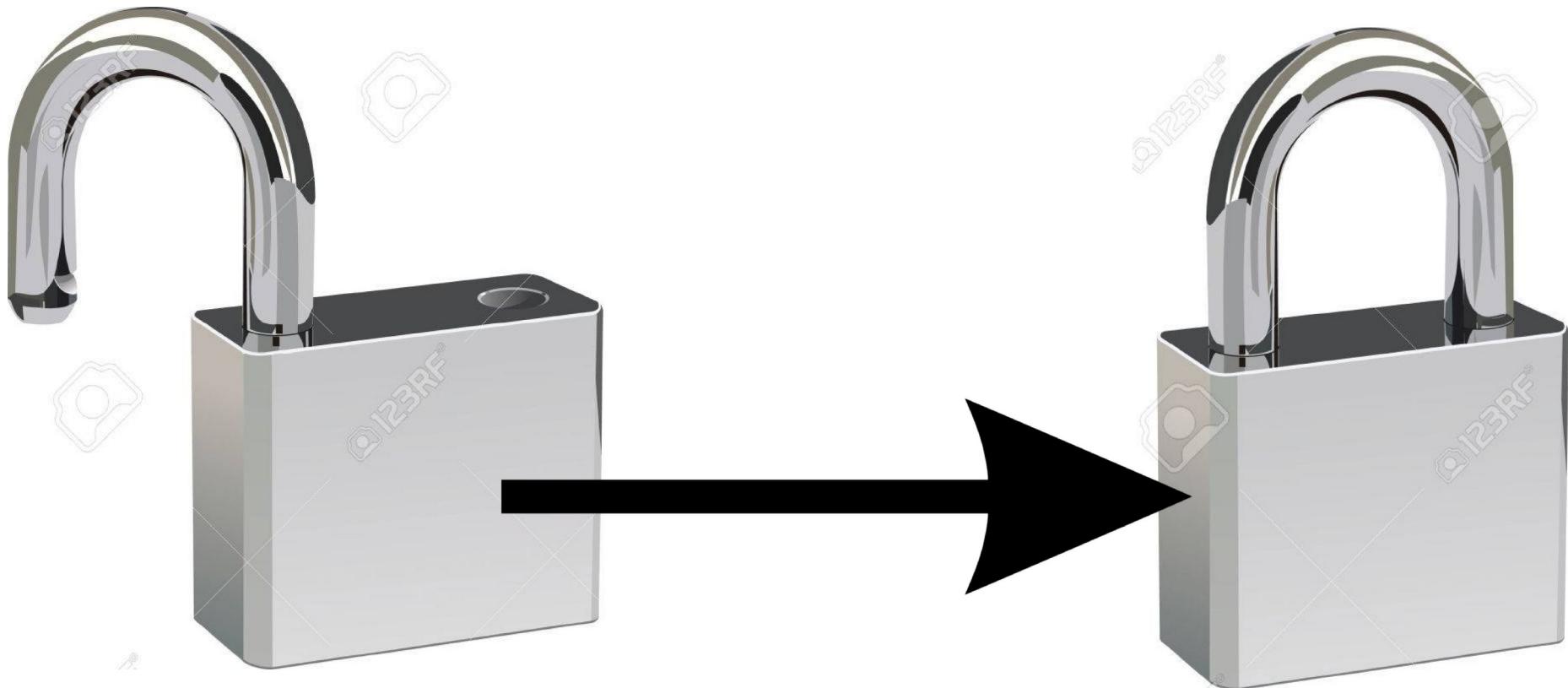
Fogalmak: state



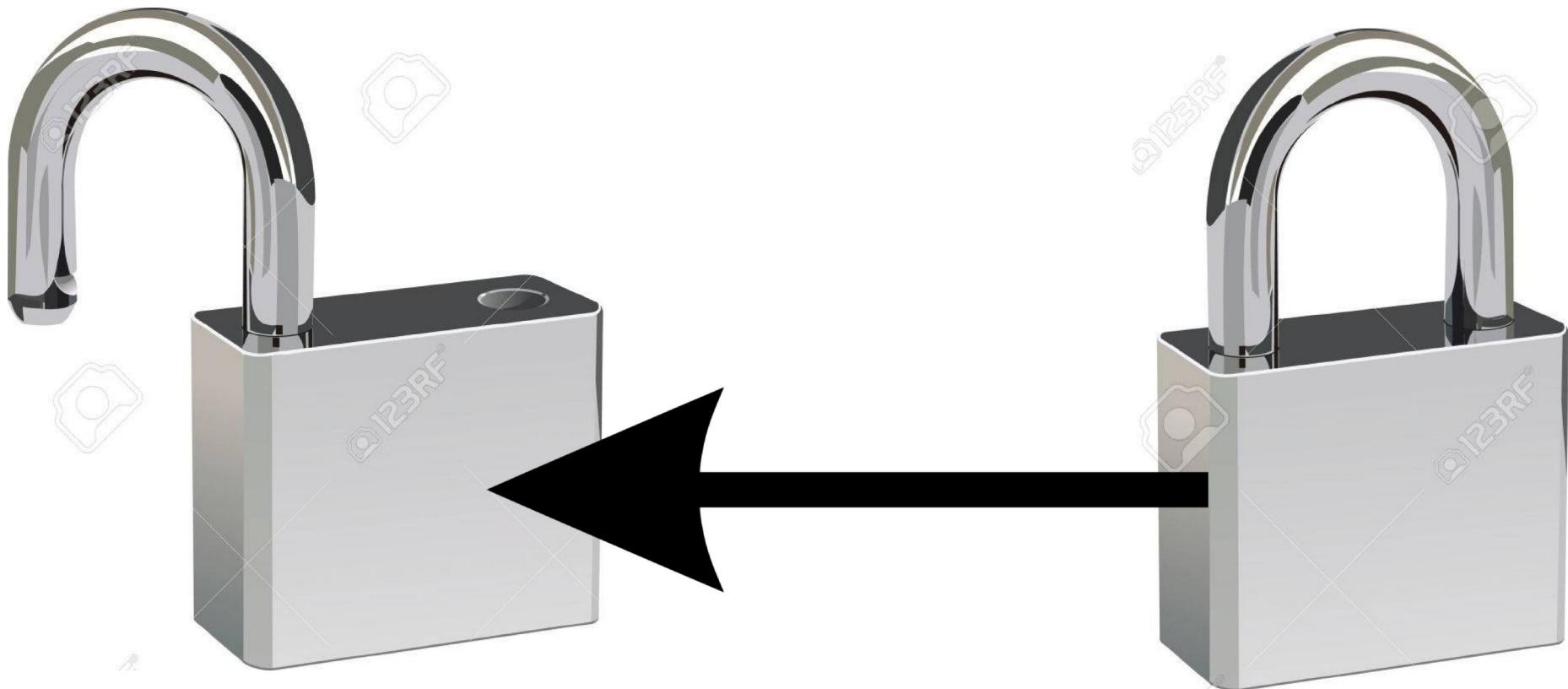
Fogalmak: state



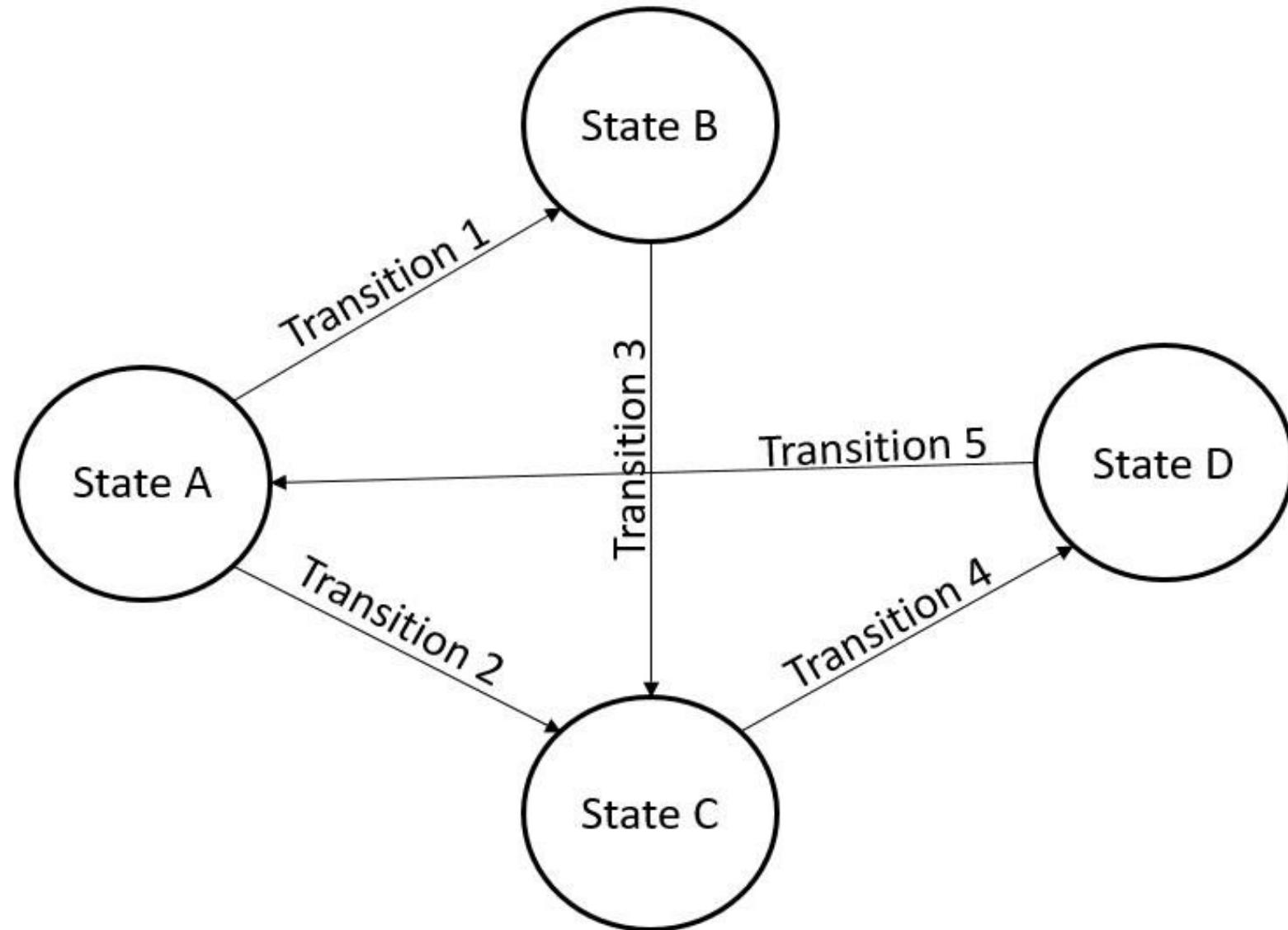
Fogalmak: transition



Fogalmak: transition



Fogalmak: state machine



State machine

- véges számú state
- aktuális state
- kezdő state
- reprezentáció: int + enum
- transition: hatására

State machine

- véges számú state
- aktuális state
- kezdő state
- reprezentáció: int + enum
- transition:  hatására

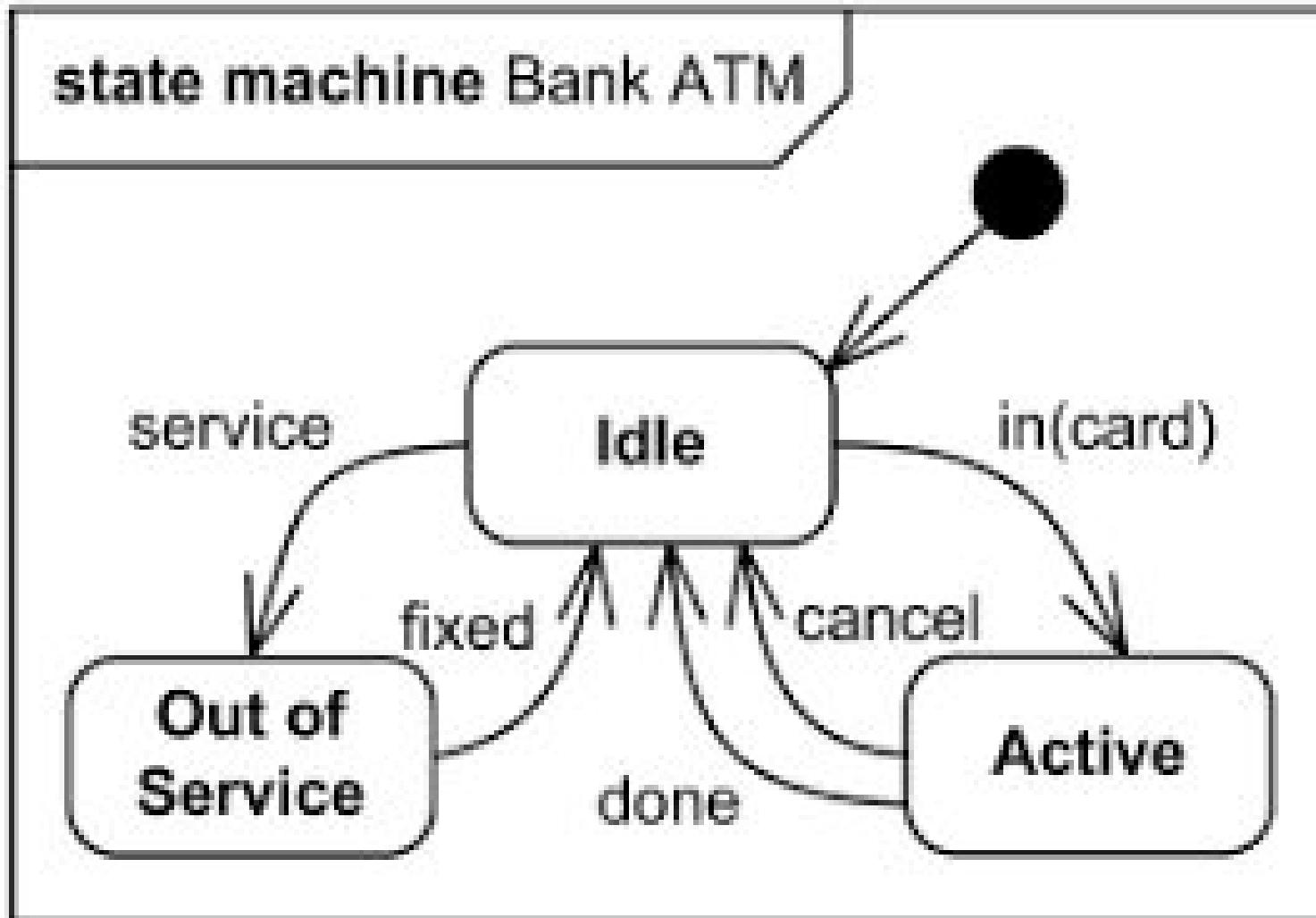
State machine

- véges számú state
- aktuális state
- kezdő state
- reprezentáció: int + enum
- transition: **event** hatására

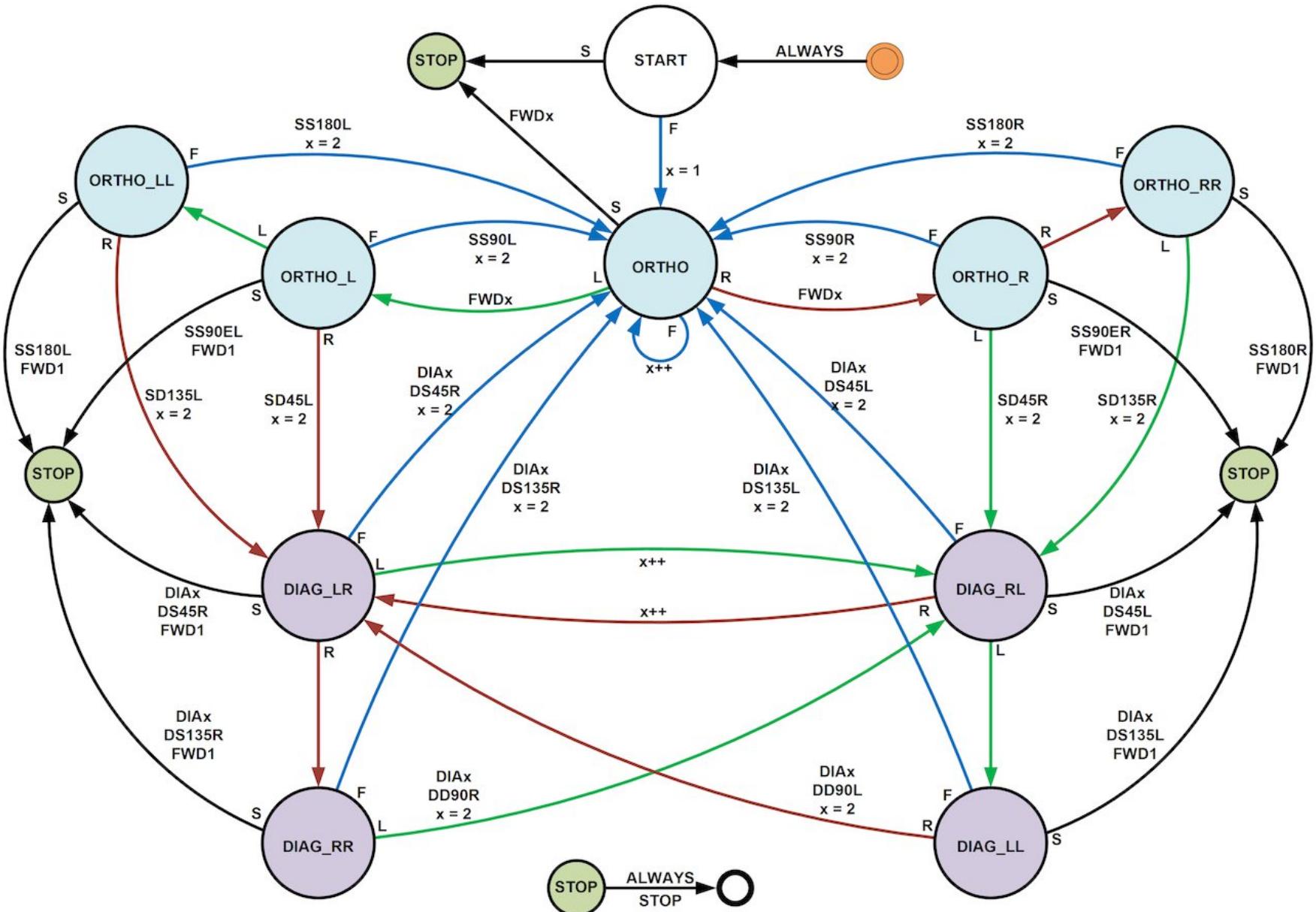
BA DUM TSSS



Egységes jelölés



Egységes jelölés

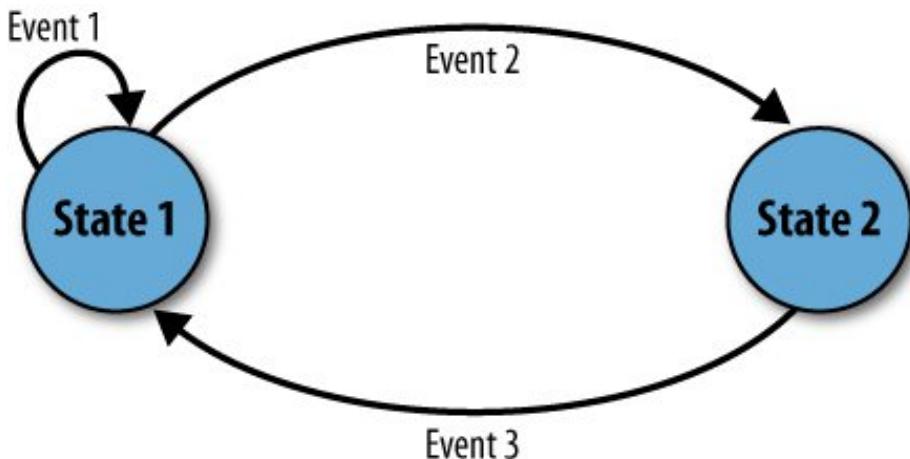


Event-driven finite state machine

- event alapok
- virtuóz event technikák
- finite state machine
- event-driven finite state machine**

Event-driven finite state machine

- state-enként eltérő az eventek kezelése (state-event matrix)
- eventek hatására akció vagy transition



```
switch (state) {  
    case ST_LOCKED:  
        switch (event) {  
            case E_BUTTON:  
                unlock();  
                state = ST_UNLOCKED;  
                break;  
        }  
        break;  
    case ST_UNLOCKED:  
        switch (event) {  
            case E_BUTTON:  
                lock();  
                state = ST_LOCKED;  
                break;  
        }  
        break;  
}
```

Event-driven finite state machine

State machine pseudo eventek:

- enter state
- leave state

Minden state önálló kis „program”.



Event-driven finite state machine

Döntési pont state:

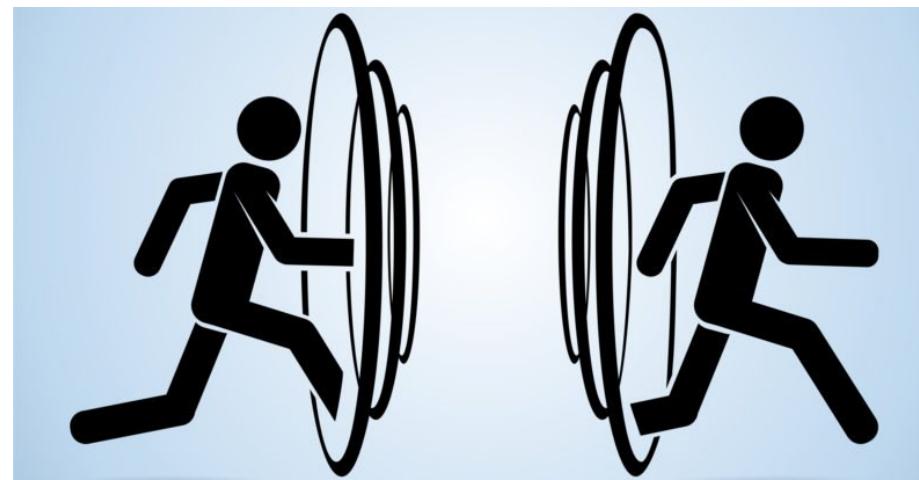
- csak az EV_STATE_ENTER van lekezelve
- azonnal továbblép másik state-re



Event-driven finite state machine

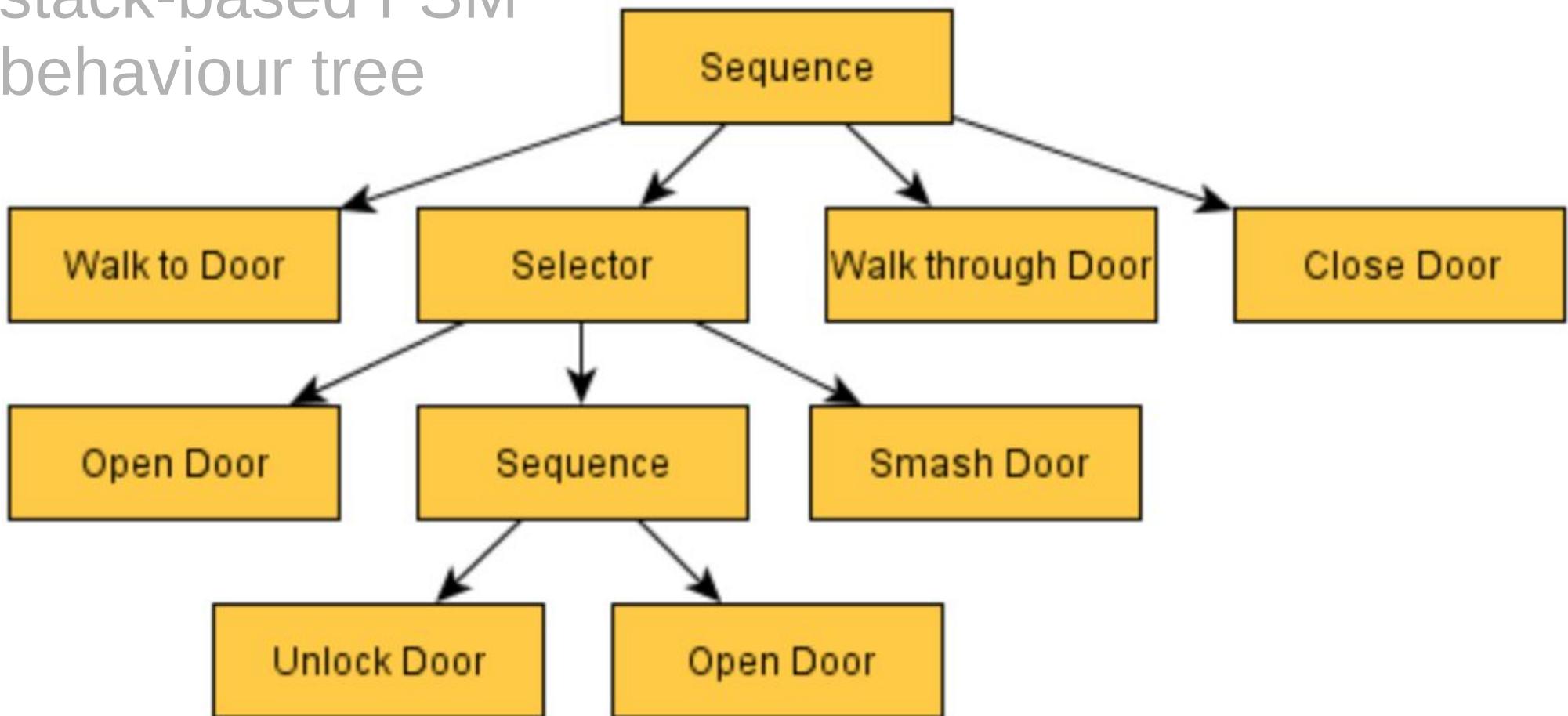
Indirekt state váltás:

- back
- előzőleg meghatározott state-re váltás
(pl. fizetési mód választás vásárlás előtt)



Összetettebb konstrukciók

- sub-state
- stack-based FSM
- behaviour tree



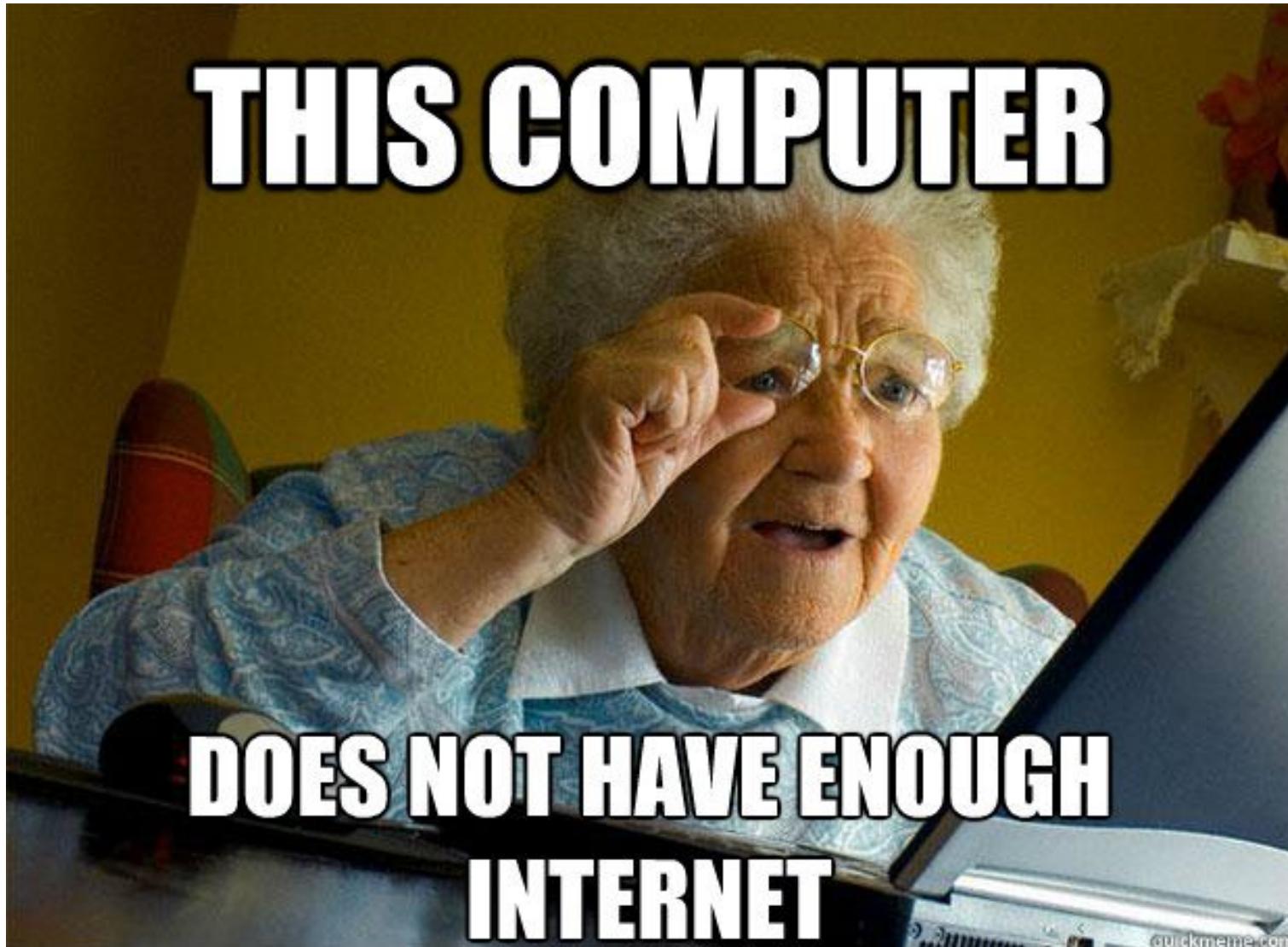
Event-driven finite state machine

Külön EDFSM modulonként



Példa: watchdog

Példa: watchdog

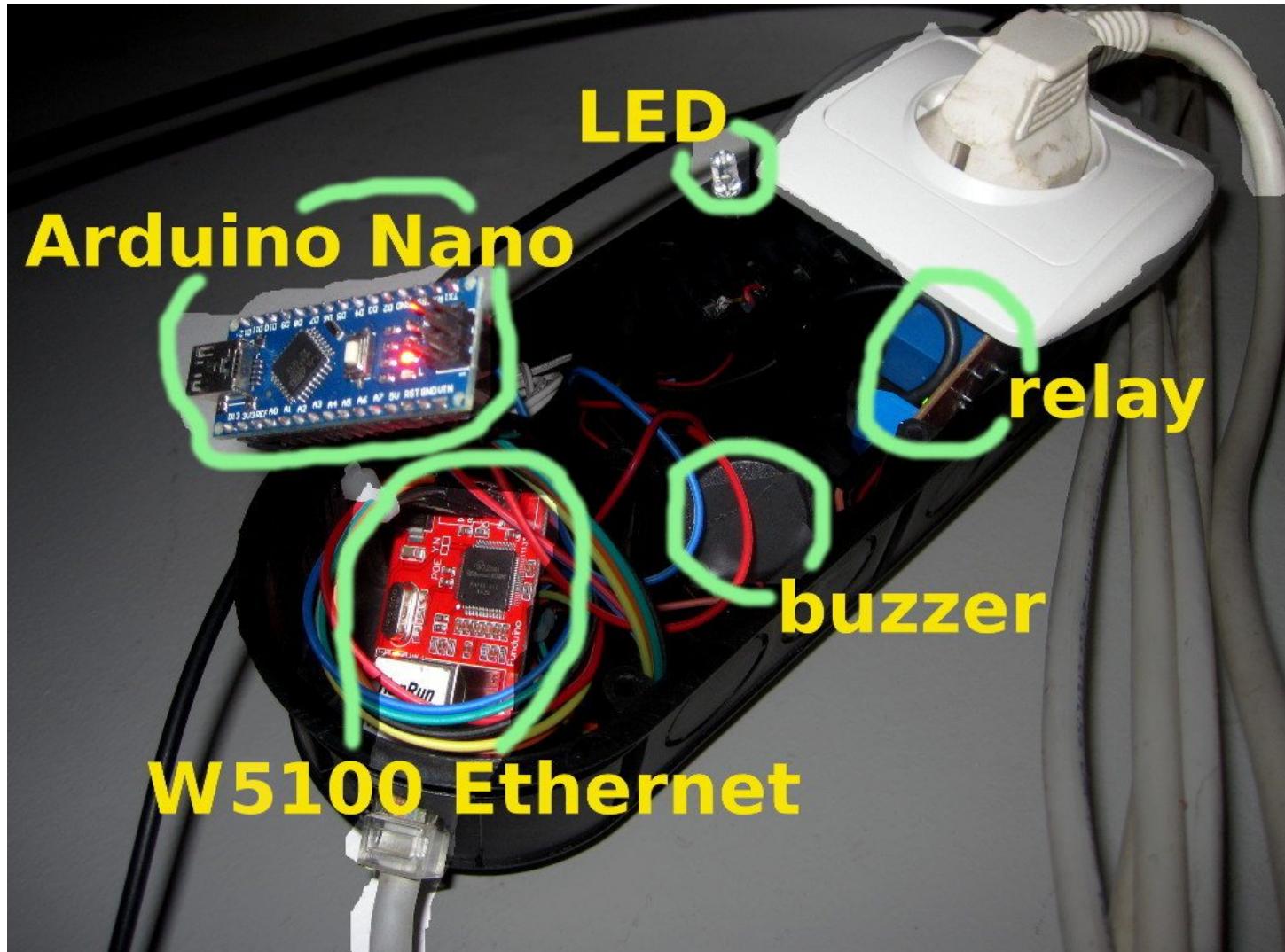


Példa: watchdog

1. *ROUTER OFF*
2. *ROUTER ON*
3. *PROFIT*



Példa: watchdog



SUCH DESIGN!
VERY IOT!



Példa: watchdog

```
# if DEBUG
    Serial.print("state: ");
    Serial.println(state);
#endif

switch (state) {
    case INIT_ETHERNET: return initEthernet();
    case RETRY_ETHERNET: return retryEthernet();
    case HTTP_CONN: return httpConnect();
    case HTTP_READ: return httpRead();
    case DELAY: return delay();
    case ALERT: return alert();
    case TURN_OFF_ROUTER: return turnOffRouter();
    case TURN_ON_ROUTER: return turnOnRouter();
    case PROTECTED_PERIOD: return protectedPeriod();
} // switch
```

Példa: watchdog

dispatcher

```
# if DEBUG
    Serial.print("state: ");
    Serial.println(state);
# endif

switch (state) {
    case INIT_ETHERNET: return initEthernet();
    case RETRY_ETHERNET: return retryEthernet();
    case HTTP_CONN: return httpConnect();
    case HTTP_READ: return httpRead();
    case DELAY: return delay();
    case ALERT: return alert();
    case TURN_OFF_ROUTER: return turnOffRouter();
    case TURN_ON_ROUTER: return turnOnRouter();
    case PROTECTED_PERIOD: return protectedPeriod();
} // switch
```

Példa: watchdog

```
turnOffRouter() {
    digitalWrite(pin,HIGH);
    setNextState( TURN_ON_ROUTER, 2 );
}

void turnOnRouter() {
    digitalWrite(pin,LOW);
    setNextState( PROTECTED_PERIOD, 1 );
}

protectedPeriod() {
    blink.play(protectedSong);
    setNextState( HTTP_CONNECT, 90 );
}
```

Példa: watchdog

event handlers

```
turnOffRouter() {
    digitalWrite(pin,HIGH);
    setNextState( TURN_ON_ROUTER, 2 );
}

void turnOnRouter() {
    digitalWrite(pin,LOW);
    setNextState( PROTECTED_PERIOD, 1 );
}

protectedPeriod() {
    blink.play(protectedSong);
    setNextState( HTTP_CONNECT, 90 );
}
```

Példa: fizetőautomata

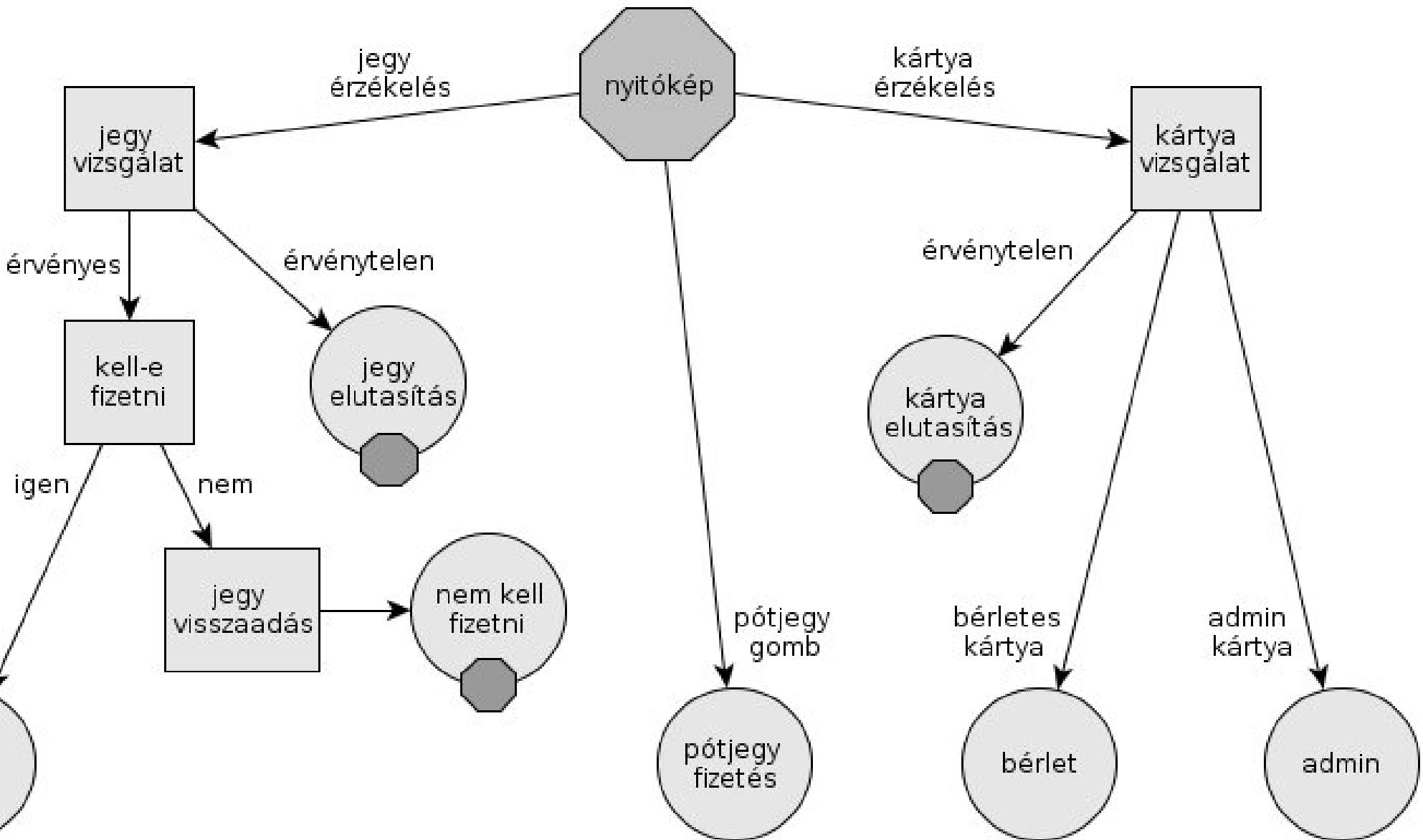
Példa: fizetőautomata



Példa: fizetőautomata

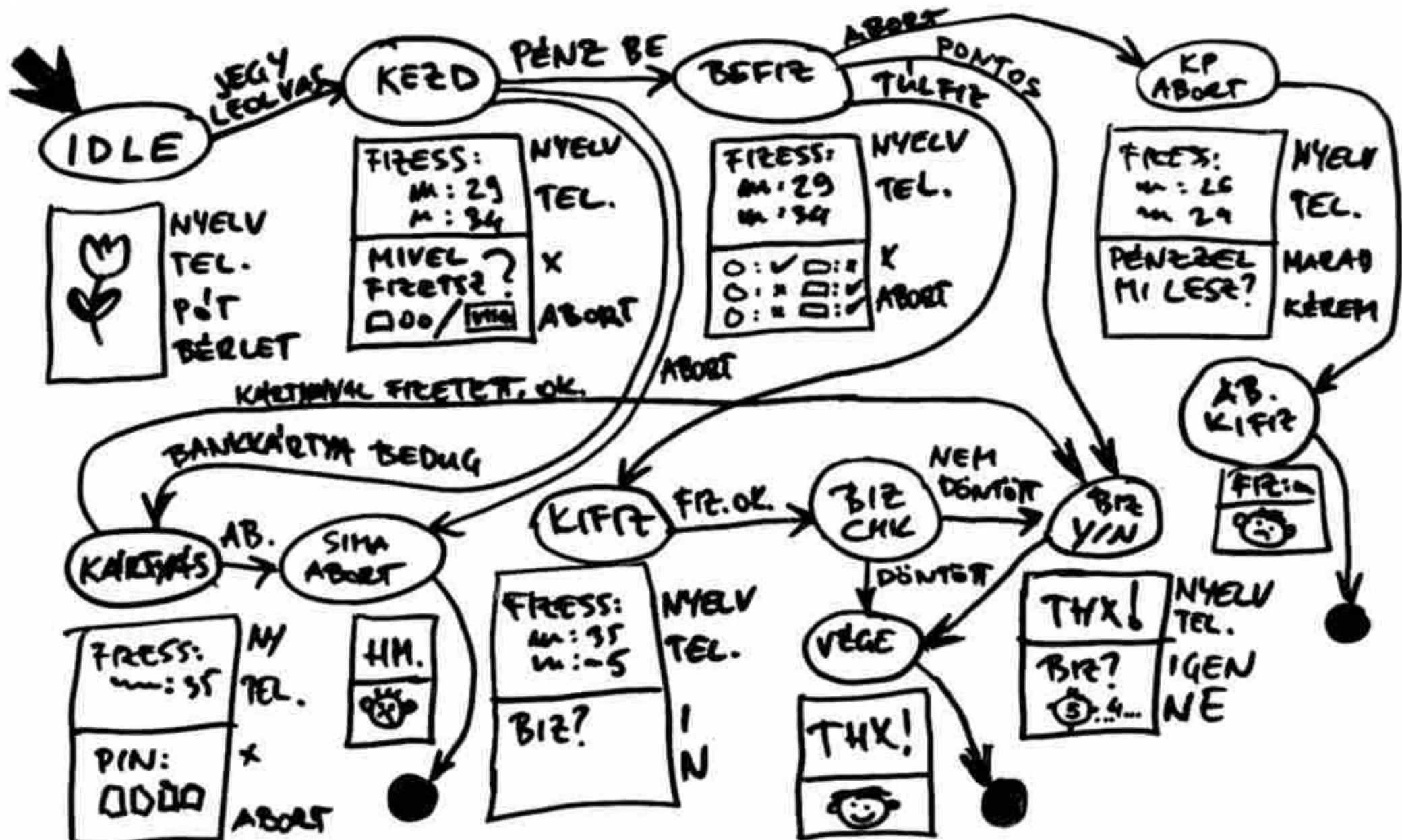


Fizetőautomata: tervezés

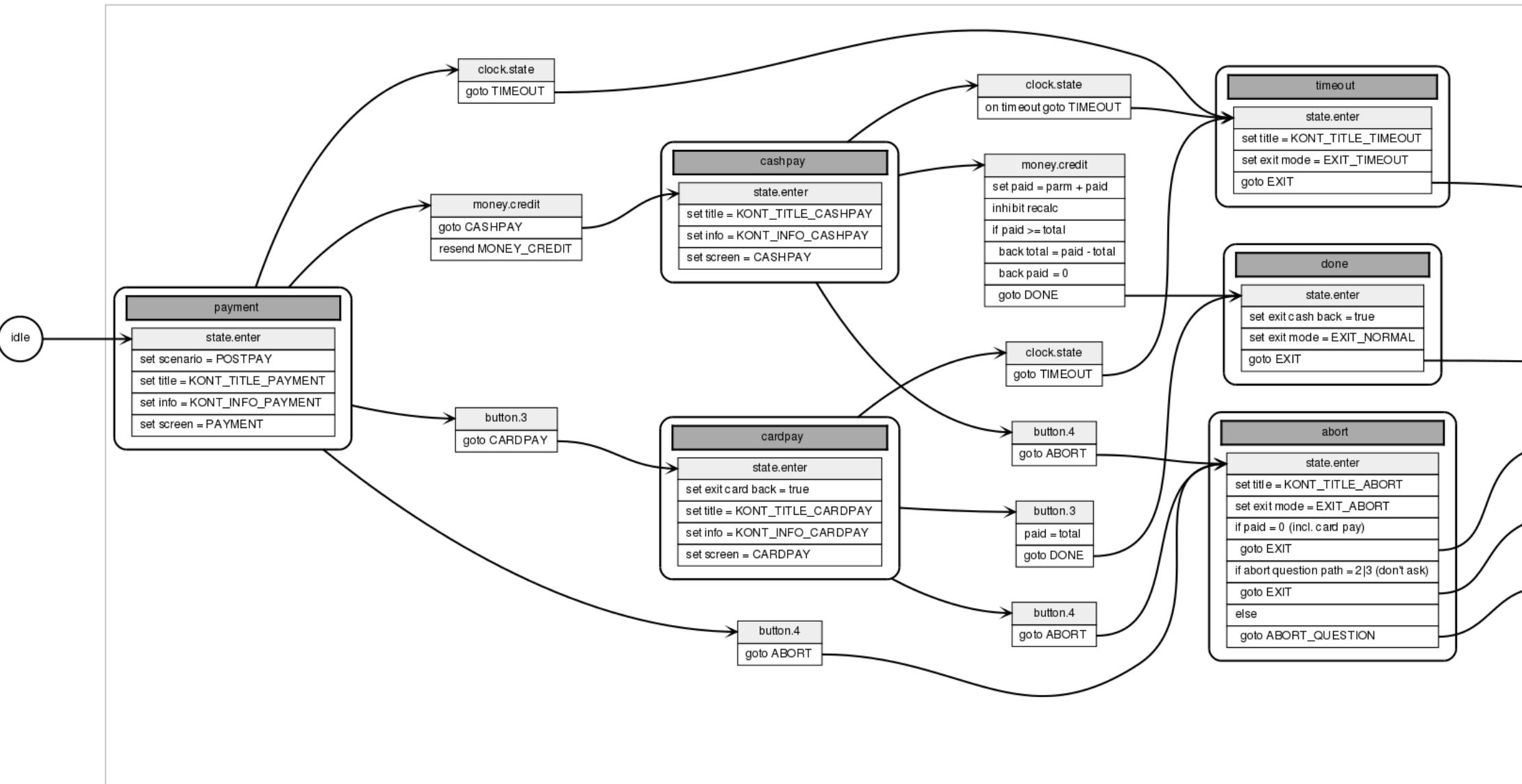




Fizetőautomata: tervezés



Fizetőautomata: dokumentálás



Fizetőautomata: dokumentálás

```
ST_GENERAL_STARTUP
ST_GENERAL_IDLE

ST_POSTPAY_SELECT_PAYMENT_MODE
ST_POSTPAY_ABORT_SIMPLE
ST_POSTPAY_ABORT_PAYBACK
K
ST_POSTPAY_CARD_PIN
ST_POSTPAY_PAY_IN
ST_POSTPAY_PAY_OFF
ST_POSTPAY_RECEIPT_PRINTING
ST_POSTPAY_FINISH THANKS
```

```
EVT_NOTE_CHECKING
EVT_NOTE_HOLD
EVT_NOTE_REJECT
EVT_NOTE_PULLOUT
EVT_NOTE_ACCEPT
EVT_NOTE_STACKING
EVT_NOTE_STACKED_IN_CASHBOX
EVT_NOTE_STORED_IN_PAYOUT
EVT_NOTE_CASHBOX_REMOVE
EVT_NOTE_CASHBOX_REPLACE
EVT_NOTE_DISPENSING
EVT_NOTE_DISPENSED
EVT_NOTE_EMPTIED
EVT_NOTE_EMPTYING
EVT_NOTE_JAM
```

```
EVT_NOTE_PAYOUT_FAILED
EVT_NOTE_OPERATION_FAILED
EVT_NOTE_IGNORED_POLL_RESPONSE
EVT_NOTE_IGNORED_CMD_RESPONSE
EVT_NOTE_UNHANDLED_POLL_RESPONSE
```

```
EVT_COIN_CREDIT
EVT_COIN_FLOATING
EVT_COIN_EMPTIED
EVT_COIN_FLOATED
EVT_COIN_PAID
EVT_COIN_PAYING
EVT_COIN_EMPTYING

EVT_COIN_PAYOUT_FAILED
EVT_COIN_OPERATION_FAILED
EVT_COIN_UNHANDLED_POLL_RESPONSE
EVT_COIN_UNHANDLED_CMD_RESPONSE
EVT_COIN_IGNORED_POLL_RESPONSE
EVT_COIN_IGNORED_CMD_RESPONSE
```

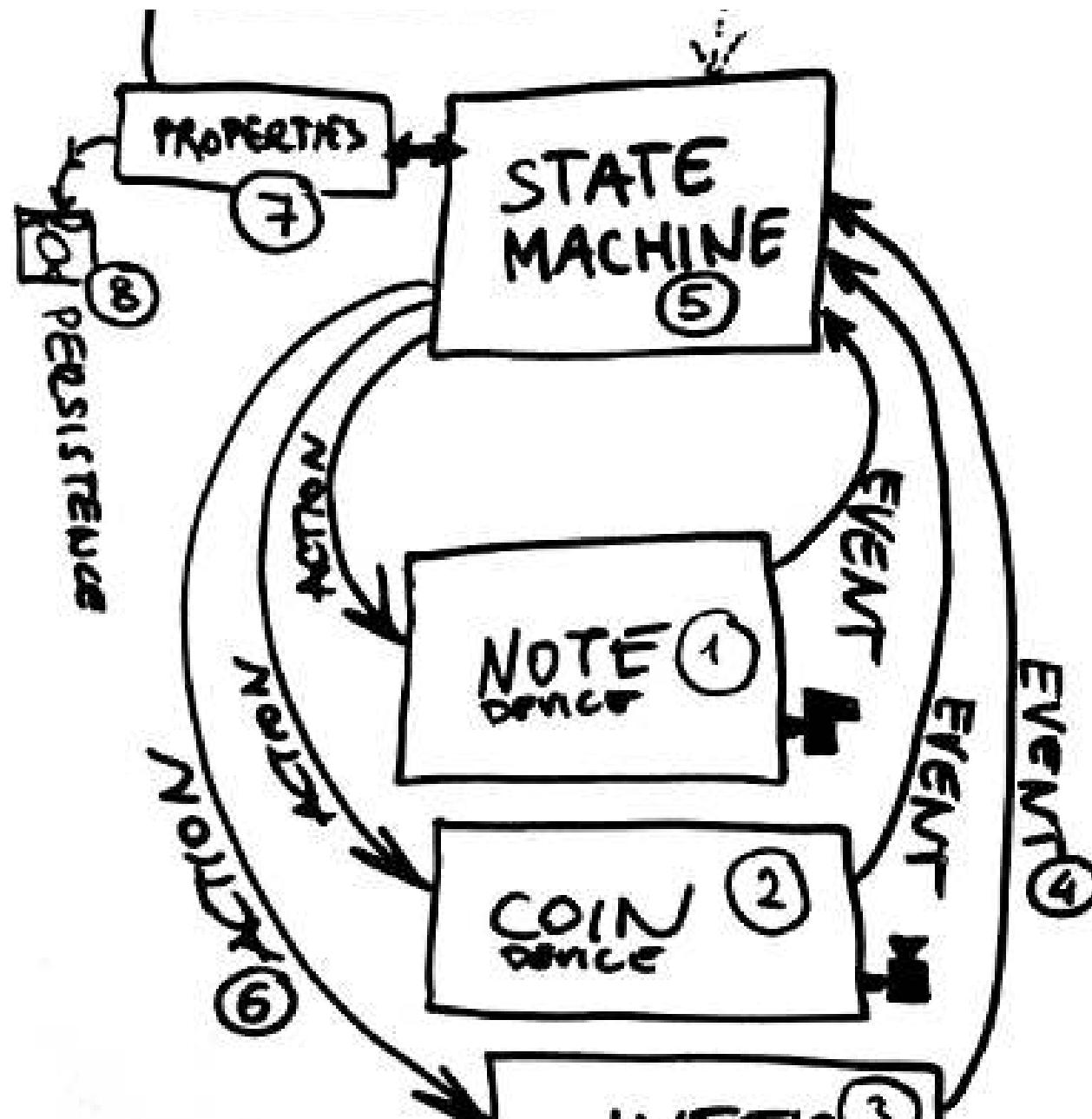
```
EVT_CHANNEL_EMPTY
EVT_CHANNEL_LOW
EVT_CHANNEL_OVERFLOW
```

```
EVT_STATE_ENTER
EVT_STATE_LEAVE
```

```
EVT_CLOCK_ACTIVITY
EVT_CLOCK_STATE
```

```
EVT_BUTTON_A
EVT_BUTTON_B
EVT_BUTTON_C
EVT_BUTTON_D
```

Fizetőautomata: rendszer



Kérdezzetek!



Kérdezzetek!





HURBA
Hungarian Robot Builders Association

köszönöm a figyelmet

ern0

All

Images

Maps

Videos

About 274.000 results (0.53 seconds)

