SDL Events

Projekt: SDLEvents

Bevezető

Az előző projektben megismerkedtünk az SDL könyvtár alapvető használatával. Ebben a projektben tovább folytatjuk az előző projektünket, eseménykezeléssel fogunk foglalkozni. Ehhez elsősorban a renderelésen kell módosítanunk.

Program bemutatása

Futtassuk le először a kódot és nézzük meg, mit csinál.

Eseménykezelés röviden

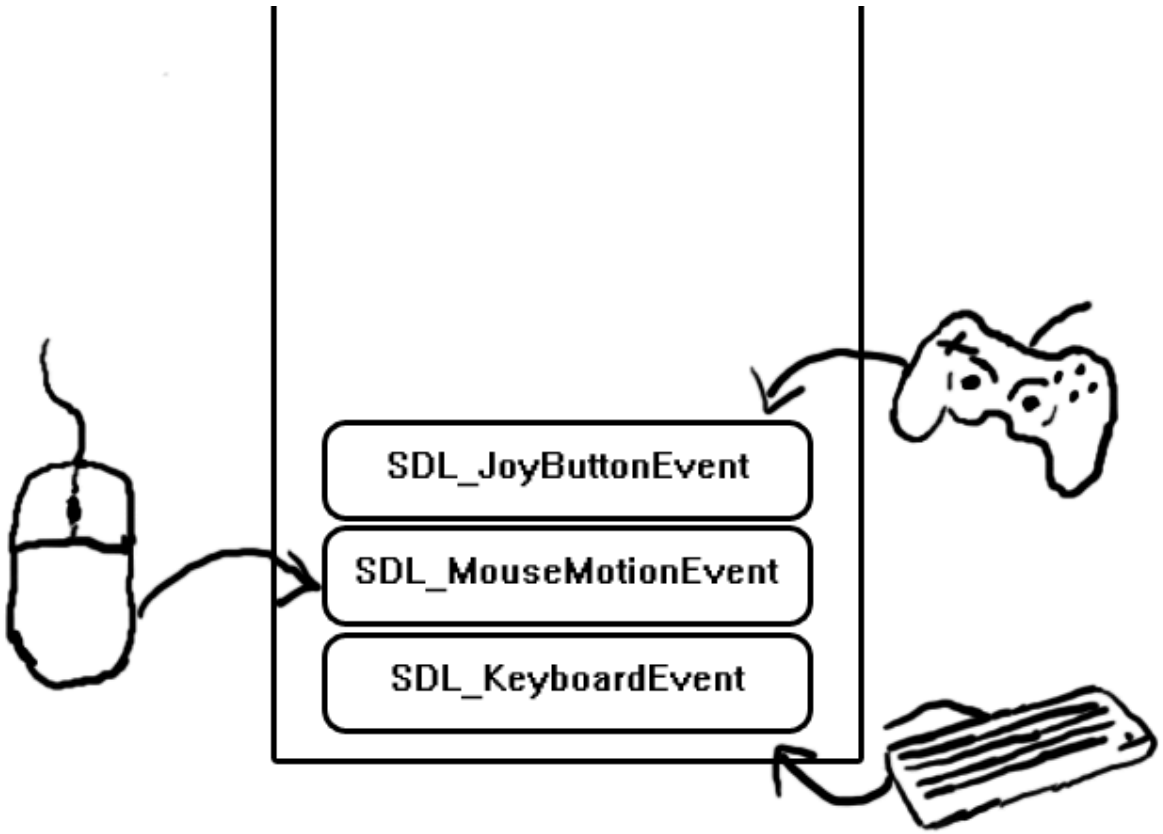
```
bool quit = false;                          // véget kell-e érjen a program futása?
float mouseX = 0.0f, mouseY = 0.0f; // egér X és Y koordinátái

while (!quit)
{
	// update() - állapot frissítése
	// render() - megjelenítés
}
```

Minden frame renderelésénél szükséges tudnunk, hogy hol van jelenleg a kurzor. Tekintsük tehát a kurzor pozícióját az állapotnak. Szintén tudnunk kell, hogy mikor lépjünk ki, ezért vegyük hozzá az állapothoz, hogy kell-e még egy frame. Tehát legyen ez a három változó az állapotunk.

Értelemszerűen ekkor a megjelenítéshez csináljuk a következőt: amíg szükséges új frame, frissítsük az állapotunkat, majd a friss állapotot jelenítsük meg.

Események kezelése

Többnyire az állapotunk frissítéséhez tudnunk kell, hogy milyen események következtek be (pl. egérmozgás, kattintás, billentyű leütés, ablak átméretezése). Ehhez az SDL biztosít egy **event queue**-t. Ez egy olyan sor, amibe belepakolja, milyen események történtek.



Ezért az állapotunk frissítéséhez kiolvassuk, hogy milyen események történtek (kiürítjük a sort), majd ezeknek megfelelően frissítjük az állapotunkat.

```
SDL_Event ev; //A feldolgozandó esemény ide kerül

while (SDL_PollEvent(&ev))
{
    switch (ev.type)
    {
    case SDL_EVENT_QUIT:
        // A felhasználó bezárja az ablakot
        quit = true;
        break;
    case SDL_EVENT_KEY_DOWN:
        // Billentyű lenyomása
    case SDL_EVENT_MOUSE_MOTION:
        // Egér mozgás
    case SDL_EVENT_MOUSE_BUTTON_UP:
        // Egérgomb felengedése
    // ....
    }
}
```

SDL_Event[1]

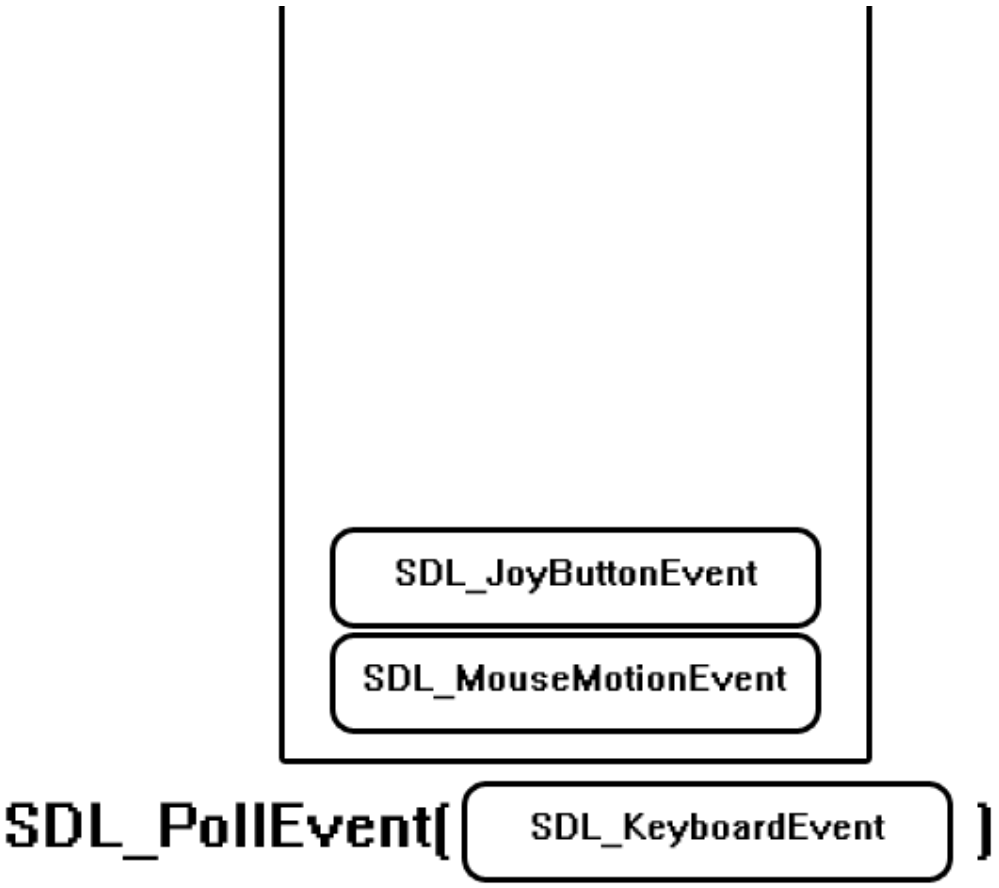
[1] A gyári események listája:
https://wiki.libsdl.org/SDL3/SDL_Event#relationships-between-event-types-and-union-members

SDL_PollEvent:

Beolvassa a változóba a sor elejéről a következő eseményt.

Ha a sor üres, hamissal tér vissza, különben igazzal.[2]

```
bool SDL_PollEvent(SDL_Event * event);
```



Renderelés

Itt már az előző projektben ismertetett függvényekkel találkozhatunk. Néhány új függvény…

```
// definiáljunk egy (mouseX, mouseY) középpontú, tengelyekkel párhuzamos oldalú
// 20x20-as négyzetet:
SDL_FRect cursor_rect;
cursor_rect.x = mouseX - 10.0f;
cursor_rect.y = mouseY - 10.0f;
cursor_rect.w = 20.0f;
cursor_rect.h = 20.0f;

SDL_SetRenderDrawColor(ren, 255, 0, 0, 255);
SDL_RenderFillRect(ren, &cursor_rect);
```

[2] https://wiki.libsdl.org/SDL3/SDL_PollEvent#return-value

SDL_FRect:

Egy téglalap adatait tárolja: bal felső sarkának koordinátái és oldalainak mérete.

SDL_RenderFillRect:

Az adott *rendering context*-ben kirajzolja a megadott téglalapot.

```
bool SDL_RenderFillRect(SDL_Renderer * renderer, const SDL_FRect * rect);
```

```
SDL_GetTicks();
```

SDL_GetTicks:

Visszatér az SDL könyvtár inicializálása óta eltelt idővel milliszekundumban.

```
Uint64 SDL_GetTicks(void);
```

Források

- képek [link]