Podstawy Programowania 1

Podstawy Grafiki 2D - część pierwsza
Biblioteka Allegro

Arkadiusz Chrobot

Zakład Informatyki

27 stycznia 2020

Plan

Wprowadzenie

Współczesne systemy komputerowe posiadają zaawansowane układy graficzne pozwalające generować zbliżone do rzeczywistości obrazy i animacje trójwymiarowe. Ten wykład będzie poświęcony prostszemu zagadnieniu jakim jest grafika dwuwymiarowa. W drugiej części wykładu zostaną przedstawione proste, lecz interesujące programy przykładowe generujące mniej lub bardziej skomplikowane obrazy i animacje. Pierwsza część wykładu będzie poświęcona bibliotece Allegro, która posłuży do pisania takich programów. Ta biblioteka, przeznaczona dla języków programowania C i C++ początkowo była wzorowana na innej, zaprojektowanej dla komputerów Atari ST. Jej głównym zastosowaniem jest tworzenie gier komputerowych, zarówno 2D, jak i 3D, dlatego oprócz funkcji obsługujących obraz i klawiaturę oferuje również podprogramy obsługi dźwięku, myszy, a nawet pozwalające tworzyć graficzne interfejsy użytkownika (GUI). W ramach wykładu zostaną jednak przedstawione jedynie jej podstawowe funkcje. Będziemy się posługiwać wersją 4.4 biblioteki Allegro. Więcej informacji na temat tej biblioteki można znaleźć na stronie: http://liballeg.org/. Stąd można też pobrać tę bibliotekę, gdyż jest to oprogramowanie dostępne na licencji typu *open source*.

Inicjacja i finalizacja

Głównym plikiem nagłówkowym, który należy włączyć do kodu źródłowego programu, aby móc się posłużyć w nim zawartością biblioteki Allegro jest plik `allegro.h`. Jeśli zamierzamy korzystać z klawiatury to dodatkowo powinniśmy włączyć plik `allegro/keyboard.h` (zapis oznacza: plik `keyboard.h` znajdujący się w katalogu `allegro`). Głównym podprogramem inicjującym działanie biblioteki jest makro `allegro_init`. Nie przyjmuje ono żadnych argumentów, ale zwraca liczbę typu `int`. Jeśli jest ona różna od zera, to znaczy, że nie udało się zainicjować pracy biblioteki. Informację o wyjątku można wypisać na ekran za pomocą funkcji `allegro_message()`, która jest wywoływana w taki sam sposób jak `printf()`. Ta funkcja w środowiskach wyposażonych w GUI tworzy okienko z wiadomością, a w systemach pracujących w trybie tekstowym drukuje informacje bezpośrednio na ekranie. Może ona być używana jednie przed wywołaniem funkcji `set_gfx_mode()`, która będzie opisana później. Opis wyjątku działania funkcji i makr z biblioteki Allegro zapisywany jest w zmiennej globalnej `allegro_error`, która jest tablicą elementów typu `char`.

Inicjacja i finalizacja

Funkcja `set_color_depth()` pozwala określić tzw. przestrzeń kolorów, czyli na ilu bitach będzie kodowana informacja o pojedynczym kolorze. Przyjmuje ona jako argument liczbę typu `int`, ale dozwolone wartości to 8, 15, 16, 24 i 32. Ostatnia wartość będzie używana w przykładowych programach. Oznacza ona, że informacja o kolorze będzie zajmowała cztery bajty. Pierwszy będzie składową czerwoną, drugi zieloną, trzeci niebieską wynikowego koloru, a czwarty to tzw. kanał alfa, którego wartość decyduje o przezroczystości. Taki format opisu koloru nazywa się RGBA. Funkcja `set_color_depth()` nie zwraca żadnej wartości.

Inicjacja i finalizacja

Funkcja `set_gfx_mode()` zajmuje się przełączeniem do określonego trybu graficznego. Przyjmuje ona pięć argumentów. Pierwszym jest jedna z pięciu stałych:

- `GFX_AUTODETECT` - funkcja próbuje włączyć tryb graficzny o określonej przez programistę rozdzielczości w trybie pełnoekranowym lub okienkowym,
- `GFX_AUTODETECT_FULLSCREEN` - funkcja próbuje włączyć tryb graficzny o określonej przez programistę rozdzielczości w trybie pełnoekranowym,
- `GFX_AUTODETECT_WINDOWED` - funkcja próbuje włączyć tryb graficzny o określonej przez programistę rozdzielczości w trybie okienkowym,
- `GFX_SAFE` - funkcja próbuje włączyć jakikolwiek tryb graficzny, który zadziała,
- `GFX_TEXT` - funkcja próbuje włączyć jakikolwiek tryb tekstowy, który zadziała.

Inicjacja i finalizacja

Dwa kolejne argumenty powinny być typu `int` i określają one rozdzielczość obrazu w docelowym trybie graficznym. Pierwszy określa ile pikseli będzie liczył obraz w poziomie, a druga ile w pionie. Dwa ostatnie argumenty również powinny być typu `int` i określają rozdzielczość tzw. ekranu wirtualnego, który jest używany jeśli program korzysta z mechanizmów animacji z biblioteki Allegro. Rozdzielczość takiego ekranu jest wyliczana na podstawie rozdzielczości ekranu rzeczywistego. Jeśli program nie korzysta z mechanizmów animacji, to te argumenty powinny mieć wartość zero. Funkcja `set_gfx_mode()` zwraca wartość typu `int`. Jeśli jest ona różna od zera, to znaczy, że przełączenie do trybu graficznego nie powiodło się. Finalizację pracy biblioteki Allegro przeprowadza bezargumentowa funkcja `allegro_exit()`, która nie zwraca żadnej wartości. W programie korzystającym z tej biblioteki należy użyć makra `END_OF_MAIN` *zaraz za* definicją funkcji `main()`. Nie przyjmuje ono żadnych argumentów i wykonuje operacje związane z obsługą różnych systemów operacyjnych, na których biblioteka Allegro może być użyta.

Organizacja obrazu

W trybie graficznym obraz na ekranie podzielony jest na punkty nazywane *pikselami*. Program używający biblioteki Allegro ma dostęp do ekranu za pomocą zmiennej globalnej o nazwie `screen`, która jest typu `BITMAP *`, gdzie `BITMAP` jest typem struktury opisującej zbiór pikseli nazywany *bitmapą*. Liczbę pikseli w obrazie widocznym na ekranie monitora określają dwie stałe:

`SCREEN_H` - liczba pikseli w pionie,

`SCREEN_W` - liczba pikseli w poziomie.

Każdy z pikseli na ekranie ma współrzędne, których składowe są liczbami naturalnymi. Punkt początkowy układu współrzędnych jest w lewym górnym rogu ekranu. Składowe poziome współrzędnych rosną od strony lewej do prawej. Wartość maksymalna dla nich to `SCREEN_W-1`. Wartości składowych pionowych rosną od góry ku dołowi. Wartość maksymalna dla nich to `SCREEN_H-1`. Punkt początkowy ma współrzędne $(0, 0)$.

Rysowanie prymitywów

Prosty element obrazu graficznego nazywa się w grafice komputerowej *prymitywem*. Najprostszym z nich jest piksel. Funkcja `putpixel()` służy do zmiany koloru pojedynczego piksela. Pierwszym argumentem tej funkcji jest wskaźnik na strukturę bitmapy (`BITMAP *`), w której należy zmienić kolor piksela. Najczęściej tym wskaźnikiem jest `screen`. Dwa kolejne argumenty są typu `int` i stanowią składową poziomą i pionową współrzędnych piksela, którego kolor ma być zmieniony. Wartości współrzędnych są liczone względem lewego górnego rogu bitmapy, który ma zawsze współrzędne $(0, 0)$. Czwarty argument jest również typu `int` i jest to kod koloru. Można go uzyskać na kilka sposobów. Najczęstszym jest zastosowanie funkcji `makecol()`, która przyjmuje trzy argumenty typu `int`. Są to odpowiednio: składowa czerwona, zielona i niebieska wynikowego koloru piksela. Pomimo posiadanego typu, wartości te powinny należeć do zakresu od 0 do 255. Wartością zwracaną przez tę funkcję jest kod koloru o typie `int`. Inny sposób nadania pikselowi koloru, to skorzystanie z palety 256 kolorów zapisanych w tablicy `palette_color`. Każdy element tej tablicy zawiera kod koloru.

Rysowanie prymitywów

Odcinek może być narysowany na obrazie za pomocą funkcji `line()`. Przyjmuje ona sześć argumentów wywołania. Pierwszy argument to wskaźnik na strukturę bitmapy. Dwa kolejne, typu `int`, to współrzędne piksela początkowego odcinka, a dwa kolejne, tego samego typu, to współrzędne piksela końcowego. Ostatni argument jest kodem koloru odcinka. Okrąg rysowany jest przez funkcję `circle()`, która przyjmuje pięć argumentów. Pierwszym jest wskaźnik na bitmapę. Pozostałe mają typ `int`. Drugi i trzeci argument to współrzędne środka okręgu, czwarty to długość promienia, a piąty jest kodem koloru. Prostokąt, w tym kwadrat, rysowany jest funkcją `rect()`. Przyjmuje ona sześć argumentów wywołania. Pierwszy jest wskaźnikiem na bitmapę. Dwa kolejne, typu `int`, to współrzędne lewego górnego rogu rysowanego prostokąta, a dwa następne, tego samego typu, to współrzędne prawego dolnego roku. Ostatni argument, też typu `int`, to kod koloru rysowanego prostokąta. Wszystkie funkcje rysujące prymitywy nie zwracają żadnych wartości.

Obsługa klawiatury

Obsługę klawiatury inicjuje bezargumentowa funkcja `install_keyboard()`. Jeśli zwróci ona wartość różną od zera, to będzie to oznaczało, że nie udało się zainicjować obsługi klawiatury. Informacje o naciśniętych klawiszach nie są bezpośrednio odczytywane z klawiatury, ale z fragmentu pamięci operacyjnej komputera, który nazywa się *buforem klawiatury*. Do sprawdzania, czy w buforze znajdują się informacje o jakimkolwiek naciśniętym klawiszu służy bezargumentowa funkcja `keypressed()`. Zwraca ona liczbę typu `int`. Jeśli jest ona zerem, to znaczy że bufor klawiatury jest pusty. Informację o naciśniętym klawiszu z bufora klawiatury zwraca bezargumentowa funkcja `readkey()`. Wartością przez nią zwracaną jest liczba typu `int`, której najmłodszy bajt zawiera kod ASCII znaku związanego z klawiszem, a bajt znajdujący się przed nim jest kodem klawisza (ang. *scancode*). Dosyć często ta funkcja jest wykorzystana do zatrzymywania działania programu do czasu naciśnięcia przez użytkownika dowolnego klawisza. Należy jednak pamiętać, że niektóre klawisze zapisują do bufora klawiatury więcej niż dwa bajty informacji, w związku z tym ta funkcja nie zawsze się zatrzyma.

Obsługa klawiatury

Aby uniknąć sytuacji opisanej pod koniec poprzedniego slajdu należy zadbać, aby `readkey()`, jeśli ma blokować wykonanie programu, była zawsze wywoływana, kiedy bufor klawiatury jest pusty. Należy zatem wywoływać przed nią bezargumentową funkcję `clear_keybuf()`, która czyści ten bufor. Stan poszczególnych klawiszy na klawiaturze można również zbadać za pomocą tablicy o nazwie `key`. Jej elementy są typu `char` i opisują stan każdego z klawiszy klawiatury. Jeśli dany element ma wartość różną od zera, to znaczy to, że klawisz związany z tym elementem jest wciśnięty. Wartości indeksów tej tablicy zostały opisane stałymi o przedrostku `KEY_`, np.: `KEY_ESC` to indeks elementu tablicy `key` odpowiadającemu klawiszowi `Escape`, a `KEY_Q` to indeks elementu związanego z klawiszem `q`. Opis innych takich stałych można znaleźć w dokumentacji na stronie, której adres został podany na początku wykładu.

Animacja

Biblioteka Allegro dostarcza mechanizmów pozwalających szybko rysować na ekranie monitora kolejne klatki obrazu przedstawiającego ruch. Minimalna częstotliwość wyświetlania klatek na ekranie, przy której mózg człowieka ma wrażenie, że widzi płynny ruch, wynosi 24 klatki na sekundę. Taki wymóg można spełnić stosując mechanizm stron biblioteki Allegro. Strona to bitmapa, na której jest rysowana pojedyncza klatka animacji. Allegro zapewnia mechanizm szybkiej zamiany takich stron. Jego działanie zostanie opisane dla przypadku, kiedy dostępne są dwie strony. Podczas gdy zawartość pierwszej jest wyświetlana na ekranie, na drugiej jest rysowana kolejna klatka animacji. Następnie te strony zamieniane są miejscami - pierwsza jest odwzorowywana na ekranie, a na drugiej jest rysowana następna klatka animacji. Proces ten powtarzany jest tak długo, jak długo trwa animacja.

Animacja

Biblioteka Allegro dostarcza kilku funkcji, które obsługują mechanizm stron. Do tworzenia bitmapy służy funkcja `create_bitmap()`, która przyjmuje dwa argumenty wywołania typu `int` określające, odpowiednio, liczbę pikseli w poziomie i w pionie tworzonej bitmapy. Zwraca ona wskaźnik na strukturę bitmapy. Tak utworzona bitmapa musi zostać usunięta przed zakończeniem programu lub przed zmianą trybu graficznego. Usuwaniem bitmap zajmuje się funkcja `destroy_bitmap()`, która jako parametr wywołania przyjmuje wskaźnik na strukturę opisującą bitmapę do usunięcia. Ta funkcja nie zwraca żadnej wartości. Aby wyczyścić bitmapę (ustawić kolor wszystkich jej pikseli na kolor, domyślnie, czarny) należy użyć funkcji `clear_bitmap()` przyjmującej jako argument wywołania wskaźnik na bitmapę. Funkcja ta nie zwraca żadnej wartości. Funkcja `show_video_bitmap()` odwzorowuje na ekranie bitmapę opisaną strukturą, na którą wskaźnik został jej przekazany jako argument wywołania. Zwraca ona liczbę typu `int`. Jeśli jej wartość jest różna od zera, to oznacza to, że odwzorowanie się nie powiodło.

Wyświetlanie tekstu

Biblioteka Allegro dostarcza kilku funkcji, które pozwalają na umieszczenie tekstu na ekranie. Tutaj zostaną przedstawione dwie z nich. Funkcja `textout_centre_ex()` przyjmuje siedem argumentów wywołania. Pierwszym jest wskaźnik na strukturę bitmapy, drugim wskaźnik na strukturę opisującą font jaki posłuży do złożenia tekstu (`FONT *`), trzecim wskaźnik na tekst do wypisania (`char *`). Dwa kolejne argumenty, typu `int`, to współrzędne piksela, który stanowi środek wypisywanego tekstu. Kolejny argument to kod koloru wypisywanego tekstu (typ `int`), a ostatni to kod koloru tła (również typ `int`). Jeśli wartość ostatniego argumentu będzie wynosiła `-1`, to tło tekstu będzie przezroczyste. Jako drugi argument opisywanej funkcji może zostać przekazana zmienna globalna `font` będąca wskaźnikiem na strukturę opisującą domyślny font używany przez bibliotekę Allegro. Funkcja `textout_centre_ex()` nie zwraca żadnej wartości.

Wyświetlanie tekstu

Funkcja `textprintf_ex()` jest kolejną, która umieszcza tekst na bitmapie. Podobnie jak funkcja opisana na poprzednim slajdzie, jako dwa pierwsze argumenty przyjmuje ona wskaźniki na struktury opisujące odpowiednio bitmapę i font. Kolejne dwa argumenty, typu `int` są współrzędnymi piksela na bitmapie, od którego funkcja ma rozpocząć wypisywanie tekstu. Następne dwa argumenty są również typu `int`. Pierwszy z nich jest kodem koloru tekstu, a drugi koloru tła. Jeśli wartość tego ostatniego wynosi `-1`, to tło tekstu będzie przezroczyste. Pozostałe argumenty wywołania tej funkcji są takie same jak dla funkcji `printf()`. Funkcja `textprintf_ex()` nie zwraca żadnej wartości.

Przykład - prosta grafika

Na poprzednich slajdach zostały zaprezentowane jedynie te elementy biblioteki Allegro, które zostaną wykorzystane w przykładowych programach zaprezentowanych na tym i następnym wykładzie. Mimo, że jest ich niewiele, to za ich pomocą można stworzyć interesujące programy wykorzystujące grafikę 2D. Jako przykład zostanie zaprezentowany na tym wykładzie program rysujący w trybie pełnoekranowym różnokolorowe okręgi do czasu naciśnięcia przez użytkownika dowolnego klawisza na klawiaturze. Zarówno współrzędne środka, jak i długość promienia oraz kolor tych okręgów są pseudolosowe.

Przykład - prosta grafika

```
#include<allegro.h>
#include<allegro/keyboard.h>
#include<stdlib.h>
#include<time.h>

#define WIDTH 1366
#define HEIGHT 768
```

Przykład - komentarz

Na początku kodu źródłowego programu umieszczone są cztery dyrektywy preprocesora włączające pliki nagłówkowe. Pierwsze dwa z nich zostały opisane na wstępnie wykładu. Plik `stdlib.h` został włączony, bo zawiera deklarację funkcji realizujących generator liczb pseudolosowych. Ostatni nagłówek zawiera funkcję `time()` pozwalającą odczytać bieżący czas jako liczbę typu `int`. Wartość ta posłuży do zainicjowania generatora liczb pseudolosowych. Zdefiniowane stałe `WIDTH` i `HEIGHT` określają odpowiednio liczbę pikseli w poziomie i w pionie obrazu wyświetlanego na monitorze w trybie pełnoekranowym, czyli rozdzielczość tego monitora.

Przykład - prosta grafika

```
int initialize(int card, int width, int height)
{
    srand(time(NULL));
    if(allegro_init()) {
        allegro_message("allegro_init(): %s\n",allegro_error);
        return -1;
    }
    if(install_keyboard()) {
        allegro_message("install_keyboard(): %s\n",allegro_error);
        allegro_exit();
        return -1;
    }
    set_color_depth(32);
    if(set_gfx_mode(card,width,height,0,0)) {
        allegro_message("set_gfx_mode(): %s\n",allegro_error);
        allegro_exit();
        return -1;
    }
    return 0;
}
```

Przykład - komentarz

Funkcja `initialize()` odpowiedzialna jest za wykonanie inicjacji generatora liczb pseudolosowych, pracy biblioteki Allegro i obsługi klawiatury. Następnie określa ona przestrzeń kolorów (32 bitowa) oraz przełącza monitor w odpowiedni tryb graficzny. Wszystko to odbywa się poprzez użycie funkcji i makra opisanych na poprzednich slajdach. Funkcja `initialize()` posiada trzy parametry typu `int`. Pierwszy to wartość określająca sterownik karty graficznej. Za ten parametr w miejscu wywołania funkcji będzie podstawiona jedna ze stałych wymienionych w opisie funkcji `set_gfx_mode()`. Dwa pozostałe określają rozdzielczość obrazu na ekranie. Warto zwrócić uwagę na dwa ostatnie argumenty wywołania funkcji `set_gfx_mode()`. Ponieważ nie będą wykorzystywane mechanizmy animacji biblioteki, to ich wartość wynosi zero. Jeśli nie powiedzie się działanie którejś z funkcji używanych przez `initialize()`, to użytkownik jest informowany o tym za pośrednictwem funkcji `allegro_message()`, która podaje opis wyjątku. Następnie wywoływana jest funkcja finalizująca działanie biblioteki Allegro i funkcja `initialize()` zwraca wartość `-1` oraz kończy działanie. Jeśli wszystkie czynności inicjujące zakończą się prawidłowo, to funkcja ta zwróci zero.

Przykład - prosta grafika

```
void draw_random_circles(BITMAP* bitmap)
{
    clear_keybuf();
    while(!keypressed()) {
        int colour = makecol(rand()%256,rand()%256,rand()%256);
        circle(bitmap,rand()%SCREEN_W,rand()%SCREEN_H,rand()%50,colour);
    }
}
```

Przykład - komentarz

Funkcja `draw_random_circles()` zgodnie ze swoją nazwą wykonuje główną czynność w programie. Nie zwraca ona żadnej wartości, ale ma jeden parametr, przez który przekazywany jest wskaźnik na strukturę opisującą bitmapę, na której mają być rysowane okręgi. Pierwszą czynnością wykonywaną wewnątrz tej funkcji jest wywołanie funkcji czyszczącej zawartość bufora klawiatury. Następnie w pętli `while` ustalany jest kolor rysowanego okręgu poprzez wylosowanie z osobna wartości z zakresu od 0 do 255 dla każdej z jego składowych i przekazanie ich do wywołania funkcji `makecol()`, która wyliczy z nich kod koloru. Ten kod jest zapisywany w zmiennej `colour`. Okrąg jest rysowany za pomocą wywołania funkcji `circle()`. Jej pierwszym argumentem wywołania jest parametr funkcji `draw_random_circles()`, który jest wskaźnikiem na strukturę opisującą bitmapę. Składowe współrzędnych środka tego okręgu są losowane z zakresu od 0 do `SCREEN_W-1` (poziom) i od 0 do `SCREEN_H-1` (pion). Długość promienia z kolei losowana jest z przedziału od 0 do 49. Oznacza to, że niektóre okręgi w ogóle nie zostaną narysowane (promień równy 0) lub będą częściowo „wystawać" poza ekran, ze względu na położenie ich środka.

Przykład - komentarz

Pętla `while`, w której wybierany jest kolor i rysowany jest okrąg, kończy się po naciśnięciu przez użytkownika dowolnego klawisza na klawiaturze.

Przykład - prosta grafika

```
int main(void)
{
    if(initialize(GFX_AUTODETECT_FULLSCREEN,WIDTH,HEIGHT))
        return -1;
    draw_random_circles(screen);
    allegro_exit();
    return 0;
}
END_OF_MAIN()
```

Przykład - komentarz

W funkcji `main()` najpierw wywoływana jest funkcja `initialize()`. Pierwszym argumentem jej wywołania jest stała `GFX_AUTODETECT_FULLSCREEN`, co oznacza, że wywoływana w ramach `initialize()`, `set_gfx_mode()` będzie próbowała włączyć pełnoekranowy tryb graficzny. Jego rozdzielczość jest określona wartościami stałych `WIDTH` i `HEIGHT`, które są przekazywane do `initialize()` jako drugi i trzeci argument jej wywołania. Następnie wywoływana jest funkcja `draw_random_circles()`, do której przekazywana jest zmienna `screen` jako argument wywołania. Oznacza to, ze wszystkie operacje graficzne wykonywane w ramach tej funkcji będą odwzorowywane bezpośrednio na ekranie monitora. Po jej zakończeniu zostanie wywołana funkcja `allegro_exit()`, która dokona finalizacji pracy biblioteki Allegro. Proszę zwrócić uwagę na użycie po definicji funkcji `main()` makra `END_OF_MAIN`.

Pytania

?

KONIEC

Dziękuję Państwu za uwagę.