

# Sprawozdanie z projektu

## 1. Wstęp

Jako projekt wybraliśmy do napisania grę snake. Tworzyliśmy grę w trzyosobowej grupie (Piotr Nowak, Szymon Pupka, Jan Przedpeński). Na samym początku założyliśmy, że napiszemy tę grę obiektowo, jednak po czasie zobaczyliśmy, że jednak nie dajemy sobie rady i kompletnie pogubiliśmy się w kodzie a masa błędów nas przytłoczyła, dlatego postanowiliśmy przerobić nasz projekt i zmieściliśmy go w klasie main. Było to duże uproszczenie, ponieważ dzięki temu nikt nie miał już problemów ze zrozumieniem kodu i praca nad grą ruszyła.






Projekt na githubie:

[https://github.com/KISiM-AGH/projekt\\_zaliczeniowy\\_it-gr03\\_nowakp\\_pupkas\\_przedpelskij](https://github.com/KISiM-AGH/projekt_zaliczeniowy_it-gr03_nowakp_pupkas_przedpelskij)





<https://github.com/inari6735/snake2>

## 2. Uporządkowanie plików gry i ustawienia

Folder z grą wygląda następująco:





 assets	24.01.2021 16:00	Folder plików	
 music	24.01.2021 15:59	Folder plików	
 Minecrafter_3	22.01.2021 16:25	Plik czcionki True ...	12 KB
 README.md	24.01.2021 15:59	Plik MD	1 KB
 snake3.cpp	24.01.2021 15:37	C++ Source	17 KB

Mamy tutaj główny plik z kodem gry, jedynie czcionka jest wrzucona luźno w folderze „Minecrafter\_3”. Następnie mamy folder assets, w którym mamy cztery podfoldery.

 background	24.01.2021 16:00	Folder plików
 coin	24.01.2021 16:00	Folder plików
 head	24.01.2021 16:00	Folder plików
 tail	24.01.2021 16:00	Folder plików

Mieszczą one grafikę tła, rzeczy do zbierania oraz głowę i ogon węża.

Jest również folder z muzyką:

 background
 beep
 nyancat
 sword

Mieści się tutaj muzyka z gry oraz efekty dźwiękowe przy zbieraniu rzeczy przez węża.

### 3. Logika gry

Wszystkie „eventy” oraz wyświetlanie umieściliśmy w pętli while, która mieści się w głównej funkcji main. Jako jedyny parametr do pętli while przekazaliśmy zaprzeczenie zmiennej „done”, która została ustawiona na false. Części węża podzieliliśmy na trzy tablice jednowymiarowe „int snakeT[50], int snakeX[], int snakeY[50]”.

```
int snakeT[50];
for (int i = 0; i <= 50; i++) {
    snakeT[i] = 0;
}
int snakeX[50], snakeY[50];
```

Ustanowiliśmy, że każda część węża to będzie kwadrat o rozmiarach 40x40px, dlatego przy poruszaniu dodawane albo odejmowane jest 40px.

```
switch (dir) {
case RIGHT: x = x + 40;
            break;
case LEFT: x = x - 40;
            break;
case UP: y = y - 40;
          break;
case DOWN: y = y + 40;
            break;
}
```

Wyświetlanie w naszym przypadku „diamentów”, które zbiera wąż było bardzo proste, ponieważ zmienne X oraz Y ustawiane są na początku na losową wartość, która nie przekracza wysokości i szerokości planszy.

```
int coinX = 40 * (rand() % 20);
int coinY = 40 * (rand() % 15);
```

A następnie wyświetlana jest grafika.

```
al_draw_bitmap(coin1, coinX, coinY, NULL);
```

Sprawdzany jest oczywiście warunek, kiedy to X i Y węży będą takie same jak współrzędne „diamentu”.

```
if (x == coinX && y == coinY) {
    score++;
    diamonds++;
    al_play_sample(MUZ, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, NULL);
    coinX = 40 * (rand() % 20);
    coinY = 40 * (rand() % 15);
    snakeT[score] = 1;
}
```

Dodaliśmy również „miecze”, które pojawiają się na planszy co jakiś czas. Funkcja sprawdzająca czy współrzędne węży i „miecza” są takie same jest prawie identyczna co ta

powyżej. Jednak gdy już wąż zbierze „miecz”, to współrzędne miecza ustawiane są na wartości -100 i -100, tak aby miecz wyświetlany był poza planszą.

```
if (x == swordX && y == swordY) {
    score += 2;
    swords += 1;
    al_play_sample(MUZ_2, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE, NULL);
    swordX = -100;
    swordY = -100;

    snakeT[score] = 2;
}
```

Ustawiona została nowa zmienna “int z = 1”, i za każdym przejściem pętli jest inkrementowana.

```
z++;
```

I jeżeli „z % 100 == 0”, to wtedy współrzędne „miecza”, zostaną wylosowane już poprawnymi wartościami losowymi. A miecz zostanie wyświetlony na planszy.

```
if (z % 100 == 0)
{
    swordX = 40 * (rand() % 20);
    swordY = 40 * (rand() % 15);
}
```

## 4. Biblioteka Allegro5

Do utworzenia gry użyliśmy biblioteki graficznej Allegro5, która obsługuje również dźwięk. Na początku pliku zostały załączone odpowiednie moduły biblioteki.

```
#include <allegro5/allegro.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_image.h>
#include <cmath>
#include <time.h>
#include <allegro5/allegro_font.h>
#include <allegro5/allegro_ttf.h>
#include <allegro5/allegro_audio.h>
#include <allegro5/allegro_acodec.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>
```

Następnie musiały zostać zainicjalizowane. I można było już używać jej funkcji.

```
al_init_primitives_addon();
al_install_keyboard();
al_init_image_addon();
al_init_font_addon();
al_init_ttf_addon();
al_install_audio();
al_init_acodec_addon();
```

Zostały ustawione takie rzeczy jak „timer”, wgrane zostały wszystkie pliki graficzne, czcionka oraz muzyka i dźwięki.

```
MUZ = al_load_sample("music/sword.mp3");
MUZ_2 = al_load_sample("music/beep.mp3");
bg_MUZ = al_load_sample("music/background.mp3");

if (!MUZ) {
    printf("Audio clip sample not loaded!\n");
    return -1;
}

al_play_sample(bg_MUZ, 1.0, 0.0, 1, ALLEGRO_PLAYMODE_LOOP, NULL);

char background_image[100] = "assets/background/background.jpg";
char head[100] = "assets/head/creeper.png";
char tail[100] = "assets/tail/tnt.png";

start:
ALLEGRO_BITMAP* head_s = al_load_bitmap(head);
ALLEGRO_BITMAP* tail_s = al_load_bitmap(tail);

ALLEGRO_FONT* font1 = al_load_font("Minecrafter_3.ttf", 15, 0);
ALLEGRO_FONT* mcfont = al_load_font("Minecrafter_3.ttf", 30, 0);
ALLEGRO_BITMAP* coin1 = al_load_bitmap("assets/coin/coin.png");
ALLEGRO_BITMAP* sword = al_load_bitmap("assets/coin/sword.png");
```

W funkcji na zdjęciu poniżej, została umieszczona usługa klawiatury.

```
if (events.type == ALLEGRO_EVENT_TIMER) {
    if (events.timer.source == VREME) times++;
    if (events.timer.source == timer) {

        al_get_keyboard_state(&keyState);
        if (al_key_down(&keyState, ALLEGRO_KEY_RIGHT) && dir != LEFT)
            dir = RIGHT;
        else if (al_key_down(&keyState, ALLEGRO_KEY_LEFT) && dir != RIGHT)
            dir = LEFT;
        else if (al_key_down(&keyState, ALLEGRO_KEY_UP) && dir != DOWN)
            dir = UP;
        else if (al_key_down(&keyState, ALLEGRO_KEY_DOWN) && dir != UP)
            dir = DOWN;
        else if (al_key_down(&keyState, ALLEGRO_KEY_A))
            score++;
        else if (al_key_down(&keyState, ALLEGRO_KEY_ENTER) && menu == true)
            menu = false, options = false, score = 1, times = 0, x = 0, y = 0;
        else if (al_key_down(&keyState, ALLEGRO_KEY_1) && menu == true) {
            menu = false;
            options = true;
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_1) && options == true) {
            options = false;
            options_1 = true;
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_BACKSPACE) && options == true) {
            menu = true;
            options = false;
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_BACKSPACE) && options_1 == true) {
            options_1 = false;
            options = true;
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_3) && options == true) {
            goto start;
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_1) && options_1 == true) {
            strcpy_s(head, "assets/head/steve.png");
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_2) && options_1 == true) {
            strcpy_s(head, "assets/head/creeper.png");
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_3) && options_1 == true) {
            strcpy_s(head, "assets/head/pig.png");
        }
        else if (al_key_down(&keyState, ALLEGRO_KEY_4) && options_1 == true) {
            strcpy_s(head, "assets/head/cow.png");
        }
    }
}
```

Menu gry zostało napisane, troszkę chaotycznie i na pierwszy rzut oka może się wydawać zbyt bardzo skomplikowane. Menu gry jest wyświetlane automatycznie przy uruchomieniu gry jak i po przegraniu.

```

if (menu) {
    x = 750, y = 550;
    for (int i = 0; i <= 50; i++) {
        snakeT[i] = 0;
    }
    al_draw_text(mcfont, al_map_rgb(100, 50, 250), 170, 100, 0,
        "Press ENTER to start");
    al_draw_text(mcfont, al_map_rgb(100, 50, 250), 170, 200, 0,
        "1.Options");
    al_draw_text(mcfont, al_map_rgb(100, 50, 250), 170, 300, 0,
        "Press ESC to exit");
    al_draw_textf(mcfont, al_map_rgb(250, 0, 250), 120, 450, 0,
        "Score: %i", score - 1);
    al_draw_textf(mcfont, al_map_rgb(250, 0, 250), 470, 450, 0,
        "Time: %i sec", timeF);
}

```

Mamy tutaj do wyboru: „Enter” – startuje nam grę, „1-Options” – po wciśnięciu klawisza 1 wyświetlane są opcje, „Esc” – po wciśnięciu klawisza ESC gra się zamyka.

Dla opcji zostały napisane dwie osobne funkcje warunkowe. Na początku funkcja warunkowa ze zmienna „options”.

```

if (options) {
    x = 750, y = 550;
    for (int i = 0; i <= 50; i++) {
        snakeT[i] = 0;
    }
    al_draw_text(mcfont, al_map_rgb(100, 50, 250), 170, 100, 0,
        "1.SKINS");
    al_draw_textf(mcfont, al_map_rgb(100, 50, 250), 170, 200, 0,
        "2.GAME SPEED(a/d): %f", FPSrate);
    al_draw_textf(mcfont, al_map_rgb(100, 50, 250), 170, 300, 0,
        "3.SAVE CHANGES", FPSrate);

    // if (al_key_down(&keyState, ALLEGRO_KEY_A))
    //     FPSrate += 1;
    // else if (al_key_down(&keyState, ALLEGRO_KEY_D))
    //     FPSrate -= 1;

    if (al_key_down(&keyState, ALLEGRO_KEY_A)) {
        FPSrate -= 1;
    }
    else if (al_key_down(&keyState, ALLEGRO_KEY_D)) {
        FPSrate += 1;
    }
}
}

```

Oraz funkcja warunkowa ze zmienną „options\_1”.

```
if (options_1) {
    x = 750, y = 550;
    for (int i = 0; i <= 50; i++) {
        snakeT[i] = 0;
    }
    al_draw_text(mcfont, al_map_rgb(0, 0, 0), 50, 50, 0,
        "Head");
    al_draw_text(mcfont, al_map_rgb(198, 204, 16), 50, 100, 0,
        "1.Steve");
    al_draw_text(mcfont, al_map_rgb(198, 204, 16), 50, 150, 0,
        "2.Creeper");
    al_draw_text(mcfont, al_map_rgb(198, 204, 16), 50, 200, 0,
        "3.Pig");
    al_draw_text(mcfont, al_map_rgb(198, 204, 16), 50, 250, 0,
        "4.Cow");
    al_draw_text(mcfont, al_map_rgb(198, 204, 16), 50, 300, 0,
        "5.Trolley");
    al_draw_text(mcfont, al_map_rgb(198, 204, 16), 50, 350, 0,
        "6.Blaze");

    al_draw_text(mcfont, al_map_rgb(0, 0, 0), 400, 50, 0,
        "Tail");
    al_draw_text(mcfont, al_map_rgb(43, 181, 80), 400, 100, 0,
        "7.TNT");
    al_draw_text(mcfont, al_map_rgb(43, 181, 80), 400, 150, 0,
        "8.Cobblestone");
    al_draw_text(mcfont, al_map_rgb(43, 181, 80), 400, 200, 0,
        "9.Sponge");
    al_draw_text(mcfont, al_map_rgb(43, 181, 80), 400, 250, 0,
        "0.Tracks");

    al_draw_text(mcfont, al_map_rgb(0, 0, 0), 250, 400, 0,
        "Background");
    al_draw_text(mcfont, al_map_rgb(189, 8, 35), 250, 450, 0,
        "Q.Standard");
    al_draw_text(mcfont, al_map_rgb(189, 8, 35), 250, 500, 0,
        "W.Nether");
    al_draw_text(mcfont, al_map_rgb(189, 8, 35), 250, 550, 0,
        "E.The End");
}
```

Obie zmienne są typu logicznego i zostały ustawione na „false”, tak aby te funkcje warunkowe, nie wyświetlały na tym etapie niczego.

```
bool options = false;
bool options_1 = false;
```

Następnie wracamy do obsługi klawiszy, tutaj są sprawdzane odpowiednie warunki.

Jeżeli jesteśmy w menu i klikniemy klawisz „1”, to zmienna menu ustawiana jest na false a zmienna options na true, co skutkuje wyświetleniem opcji i zakończeniem działania funkcji warunkowej ze zmienną menu.

```
else if (al_key_down(&keyState, ALLEGRO_KEY_1) && menu == true) {  
    menu = false;  
    options = true;  
}
```

W opcjach mamy trzy wybory „Skins”, czyli graficzne urozmaicenie naszego węża, tak aby użytkownik miał wybór, „Game speed” – dzięki temu możemy ustawiać szybkość poruszania się węża, polega to na tym, że wartość dla „timer” jest zwiększana albo zmniejszana, co skutkuje szybkością poruszania się węża. Oraz opcja „Save changes”, która zapisuje zmiany. Trzeba było zastosować taką opcję, ponieważ kod jest wykonywany non stop w pętli while, przez co zmiana wartości „timer” jak i zmiana skinów zmienia wartości zmiennych, jednak są one umieszczone poza pętlą while, i dlatego nie zapisuje zmian. Do opcji „Save changes”, została użyta funkcja „goto”. Na internecie można wyczytać, że jest to niedobra praktyka i nie powinno się tego używać, jednak w tym przypadku było to najprostsze rozwiązanie i okazało się skuteczne.

```
start:  
ALLEGRO_BITMAP* head_s = al_load_bitmap(head);  
ALLEGRO_BITMAP* tail_s = al_load_bitmap(tail);  
  
ALLEGRO_FONT* font1 = al_load_font("Minecrafter_3.ttf", 15, 0);  
ALLEGRO_FONT* mcfont = al_load_font("Minecrafter_3.ttf", 30, 0);  
ALLEGRO_BITMAP* coin1 = al_load_bitmap("assets/coin/coin.png");  
ALLEGRO_BITMAP* sword = al_load_bitmap("assets/coin/sword.png");  
  
ALLEGRO_BITMAP* background = al_load_bitmap(background_image);  
ALLEGRO_TIMER* timer = al_create_timer(1.0 / FPSrate);  
ALLEGRO_TIMER* frameTimer = al_create_timer(1.0 / frameFPS);  
ALLEGRO_TIMER* VREME = al_create_timer(1);  
ALLEGRO_KEYBOARD_STATE keyState;  
  
ALLEGRO_EVENT_QUEUE* event_queue = al_create_event_queue();  
al_register_event_source(event_queue, al_get_keyboard_event_source());  
al_register_event_source(event_queue, al_get_timer_event_source(timer));  
al_register_event_source(event_queue, al_get_timer_event_source(frameTimer));  
al_register_event_source(event_queue, al_get_timer_event_source(VREME));  
al_register_event_source(event_queue, al_get_display_event_source(display));  
  
al_start_timer(timer);  
al_start_timer(frameTimer);
```

Start zostało umieszczone w tym miejscu kodu, gdzie zaczyna się ustawianie zmiennych z grafiką oraz timer. Gdy w opcjach wybierzemy „Save changes”, to kod zaczyna się wykonywać z miejsca „start” i zmienną zostają zmienione.

```
else if (al_key_down(&keyState, ALLEGRO_KEY_3) && options == true) {  
    goto start;  
}
```



Mamy jeszcze opcję „Skins”, która działa analogicznie. Za to odpowiedzialna jest zmienna `options_1`. Gdy `options = true` i wciśniemy klawisz „1”, to `options = false`, a `options_1 = true` co skutkuje wyświetlaniem menu skinów.

```
else if (al_key_down(&keyState, ALLEGRO_KEY_1) && options == true) {  
    options = false;  
    options_1 = true;  
}
```

Gdy chcemy się cofać w opcjach wystarczy klikać klawisz „Backspace”.

```
else if (al_key_down(&keyState, ALLEGRO_KEY_BACKSPACE) && options == true) {  
    menu = true;  
    options = false;  
}  
else if (al_key_down(&keyState, ALLEGRO_KEY_BACKSPACE) && options_1 == true) {  
    options_1 = false;  
    options = true;  
}
```

## 5. Ogólny wygląd gry:





