ЛПРС2 Лаб

Напредна графика

верзија 1.2

21. март 2021.

Увод

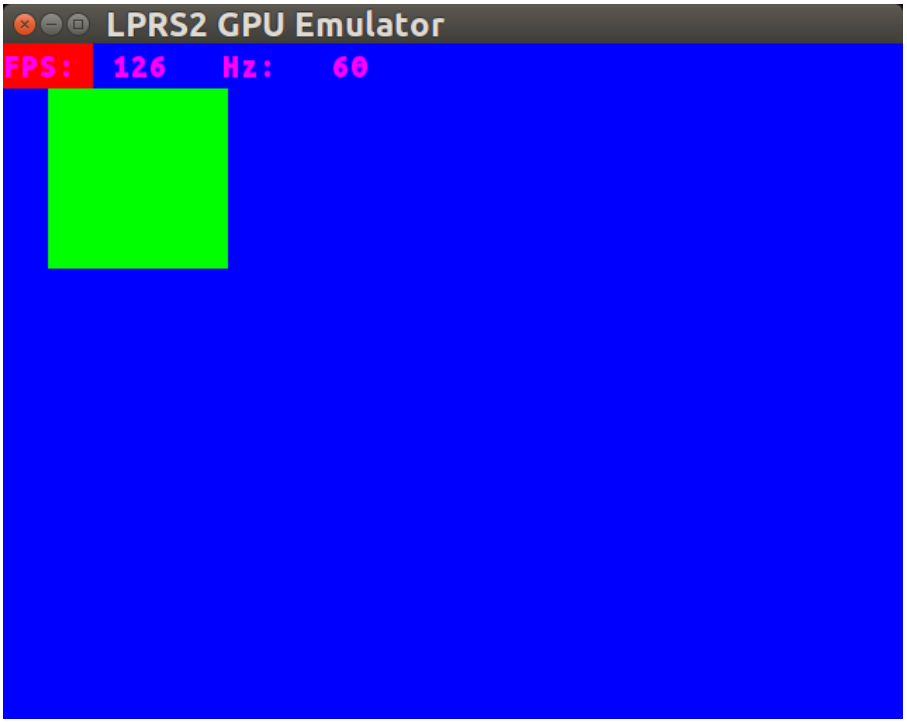
Држите отворен PDF претходне вежбе, пошто је у њему инструкције за пођеривање, меморијске мапе, и слично. Овде ћемо само увести пар нових концепата да будете спремни за пројекат, неку игрицу, стога помно испратите вежбу.

Напредни модови

Покренути програм:

```
./waf build run --app=advanced_modes
```

Када се покрене програм отвориће се прозор са изгледом као на Слици 1: У овој вежби



Слика 1: 4-битни индексни мод

је дан пример са *IDX4* и *RGB333* модом. Мењањем овог макроа

```
56 #define IDX4_0_RGB333_1 1
```

бира се код који корисит један од ова два мода. Реализована је слична апликација као и у претходној вежби, са правоугаоником и квадратом што трче около. Пошто сада користимо паковани мод за *IDX4*, сва геометрија је поравната на 8 пиксела, односно изабрано је да 8 буде јединица тј. квант геометрије. Усвојена је конвенција да сви промењиве и константе тј. поља и макрои у имену имају 8 ако је у јединицама од по 8 пиксела, тј. 1 ако је у јединицама по 1 пиксел, како је описано већ овде:

```
60 // Suffix 8 means that it is in units of 8 pixels.
61 // For example if RECT_H8 is 2,
62 // that means that height of rectangle is 2*8 = 16 pixels.
63 #define STEP8 1
64 #define RECT_H8 2
65 #define RECT_W8 4
66 #define SQ_A8 8
```

То практично значи да ширина правоугаоника одређена преко `RECT_W8` није 4 пиксела већ 32. Квант од 8 пиксела је изабран, јер у пакованом *IDX4* моду се у једној речи преноси 8 пиксела.

Следећи излист приказује начин руковања бафера у *RGB333* када је квант 8 пиксела.

```
210         // Red rectangle.
211         for(
212             uint16_t r1 = gs.rect8.y*8;
213             r1 < (gs.rect8.y+RECT_H8)*8;
214             r1++
215         ){
216             for(
217                 uint16_t c1 = gs.rect8.x*8;
218                 c1 < (gs.rect8.x+RECT_W8)*8;
219                 c1++
220             ){
221                 uint32_t idx = r1*SCREEN_RGB333_W + c1;
222                 unpack_rgb333_p32[idx] = 0007;
223             }
224         }
```

Овај бафер није пакован, тако да и даље је по једној речи један пиксел. Међутим, пошто је геометрија, описана промењивама из `gs` структуре и константама дефинисаним макроима у јединицама по 8 пиксела, потребно их је помножити са 8. Фор петље даље раде у јединицама од по 1 пиксел. Приметити коришћење октално бројевне представе, овде конкретно `0007`, где свака цуфра оредставља једну боју [1].

Следећи излист приказује начин рада у пакованом *IDX4* моду.

[1] Број `0007`, који овде представља црвену боју, не треба мешати са бројем најпознатијег Кинеског тајног агента (којих има далеко више него Енглеских па је зато број четвороцифрен), без обзира на подударност у боји заставе исте државе. Свака коенциденција је случајна.

```
        // Red rectangle.
        for(
            uint16_t r1 = gs.rect8.y*8;
            r1 < (gs.rect8.y+RECT_H8)*8;
            r1++
        ){
            for(
                uint16_t c8 = gs.rect8.x;
                c8 < gs.rect8.x+RECT_W8;
                c8++
            ){
                uint32_t idx = r1*SCREEN_IDX4_W8 + c8;
                pack_idx4_p32[idx] = 0x11111111;
            }
        }
```

Редови су као и до сада у јединицама од по 1 пиксел. Колоне су у јединицама од по 8. Конкретно за квадрат, који има `RECT_H8` постављено на 2 и `RECT_W8` постављено на 4, у сваком реду ће бити кроз 4 32-битне речи уписана 32 пиксела, и тако 16 редова. Овакан приступ изгледа ограничавајућ јер се не може радити са једним пикселом. Међутим, уз мало труда се може адаптирати алгоритам да подржи и не 8-пикселне геометрије. За неке игрице (рецимо шах), 8-пикселна геометрија је сасвим задовољавајућа.

Оно што се да приметити је да сада је број боја већи, али је то утицало на мању резолуцију слике.

Спрајтови

Покретите програм са:

```
./build/sprites
```

Покретањем програма добија се приказ као на Слици 2: У питању је штоперица. Штоперица наиме користи спрајтове (енгл. Sprites). Спрајтови се чувају у C датотеку, који има своје заглавље. Ова две датотеке се генеришу путем `img_to_src.py`, од слика које се налазе у фасцикли `images/`, што овде аутоматски одради `wscript` скрипта приликом компајлирања [2]. Овакав приступ се користи јер иначе на наменској платформи није могуће покренути отворити датотеку а и библиотеке за декодовање `png` а тек `jpg` формата су превелике. Укључивањем заглавља у програм, могуће је користити спрајтове.

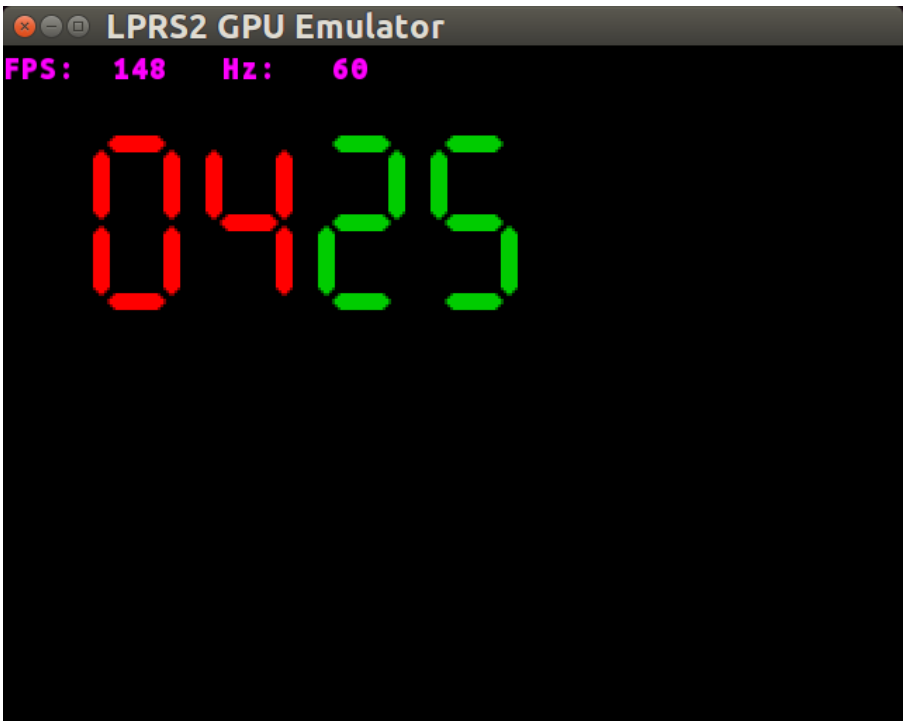
```
#include "sprites_idx4.h"
```

За сваки спрајт се чува његова ширина (суфикс `__w`) и висина (суфикс `__h`) као и низ са пикселима (суфикс `__p`). Такође се чува и палета, ако је у питању индексни мод.

Палета се просто и једноставно само копира.

```
    // Copy palette.
    for(uint8_t i = 0; i < 16; i++){
        palette_p32[i] = palette[i];
    }
```

[2] Овде се може извући интересантна поука. Иако су многи студенти, и то бољи, сад већ годинама, правили и поправљали сличне скрипте, па притом имали и више времена од аутора горепоменуте скрипте, опет скрипте студенске скрипте нису биле на нивоу. Поука за вас студенте: морате још да вежбате.



Слика 2: Штоперица са коришћењем спрајтова

Спрајтови се исцртавају путем функције `draw_sprite`. Ова функција као параметре узима низ пиксела, ширину и висину изворишног спрајта, као и одредишне позиције на екрану тј. у баферу.

```
100 static void draw_sprite(
101     uint32_t* src_p,
102     uint16_t src_w,
103     uint16_t src_h,
104     uint16_t dst_x,
105     uint16_t dst_y
106 ) {
```

Обратити пажњу артиметику за рачунање индекса (`src_idx` и `dst_idx`):

```
114     for(uint16_t y = 0; y < src_h; y++){
115         for(uint16_t x8 = 0; x8 < src_w8; x8++){
116             uint32_t src_idx = y*src_w8 + x8;
117             uint32_t pixels = src_p[src_idx];
118             uint32_t dst_idx =
119                 (dst_y+y)*SCREEN_IDX4_W8 +
120                 (dst_x8+x8);
121             pack_idx4_p32[dst_idx] = pixels;
122         }
123     }
```

Ово је од круцијалног значаја да се разуме за успешан пројекат. Овде се копира цео спрајт, тако да је аритметика мало лакша.

У овом програму, функција `draw_sprite` се користи тако што се на основу цифре штоперице индексира одговарајући низ пиксела, док су ширина и висина бетониране на 32 и 64, што је исто за све спрајтове, док се помера `x` одредишна координада за сваку цифру:

```
        // Draw digits of stopwatch.
        draw_sprite(
            red__p   [gs.digits[3]], 32, 64, 32+(3-3)*40, 32
        );
        draw_sprite(
            red__p   [gs.digits[2]], 32, 64, 32+(3-2)*40, 32
        );
        draw_sprite(
            green__p[gs.digits[1]], 32, 64, 32+(3-1)*40, 32
        );
        draw_sprite(
            green__p[gs.digits[0]], 32, 64, 32+(3-0)*40, 32
        );
```

Користе се низови низова пиксела, ради лакшег приступа одговарајућем спрајту:

```
uint32_t* red__p[16] = {
    red_0__p, red_1__p, red_2__p, red_3__p,
    red_4__p, red_5__p, red_6__p, red_7__p,
    red_8__p, red_9__p, red_a__p, red_b__p,
    red_c__p, red_d__p, red_e__p, red_f__p
};
uint32_t* green__p[16] = {
    green_0__p, green_1__p, green_2__p, green_3__p,
    green_4__p, green_5__p, green_6__p, green_7__p,
    green_8__p, green_9__p, green_a__p, green_b__p,
    green_c__p, green_d__p, green_e__p, green_f__p
};
```

Ови спрајтови вам могу добро доћи у издради пројекта.

Анимације

Покретите програм са:

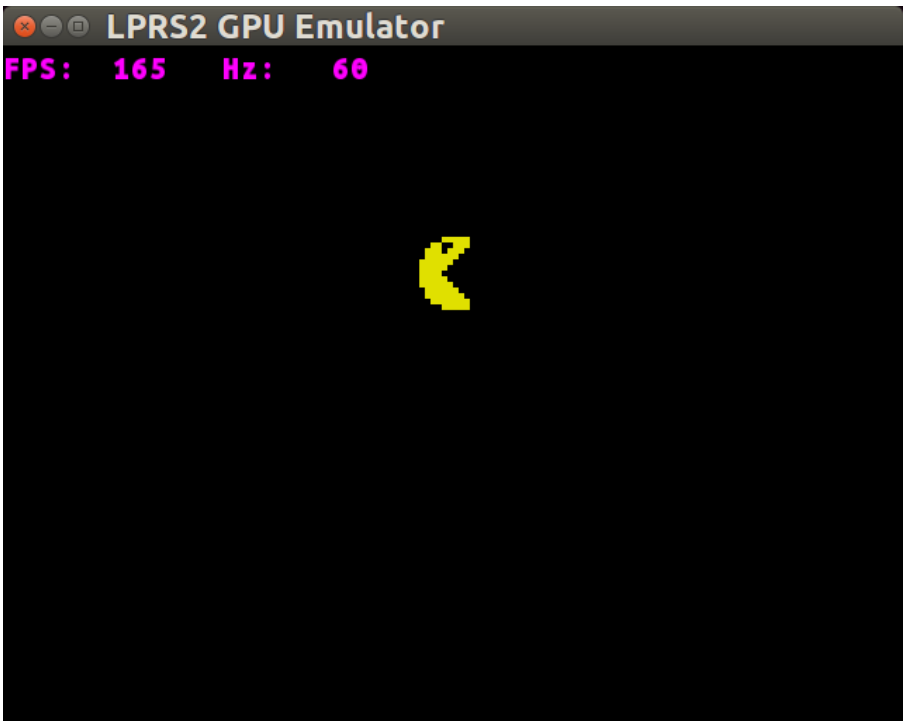
```
./build/sprite_anim
```

Приликом притиска на десну стрелицу, Pacman ће се кретати налево и зобати све (тј. ништа) пред собом, као на Слици 3.

Наиме Pacman има анимацију са 5 стања:

```
typedef enum {
    PACMAN_IDLE,
    PACMAN_OPENING_MOUTH,
    PACMAN_WITH_OPEN_MOUTH,
    PACMAN_CLOSING_MOUTH,
    PACMAN_WITH_CLOSED_MOUTH
} pacman_anim_states_t;
```

При мировању Pacman је у



Слика 3

```
74  PACMAN_IDLE,
75      PACMAN_OPENING_MOUTH,
76      PACMAN_WITH_OPEN_MOUTH,
77      PACMAN_CLOSING_MOUTH,
78      PACMAN_WITH_CLOSED_MOUTH
79  } pacman_anim_states_t;
80
81
82
83  typedef struct {
84      pacman_anim_states_t state;
85      uint8_t delay_cnt;
86  } pacman_anim_t;
87
88
89
90  typedef struct {
91      point_t pos;
92      pacman_anim_t anim;
93  } pacman_t;
94
95
96
97
98
99
100 typedef struct {
101     pacman_t pacman;
102 } game_state_t;
103
104
105
106
107
108
```

стању. Када се држи притиснут тастер, осим што ће се Pacman померити, такође ће аутомат изаћи из

```
PACMAN_IDLE,
    PACMAN_OPENING_MOUTH,
    PACMAN_WITH_OPEN_MOUTH,
    PACMAN_CLOSING_MOUTH,
    PACMAN_WITH_CLOSED_MOUTH
} pacman_anim_states_t;



typedef struct {
    pacman_anim_states_t state;
    uint8_t delay_cnt;
} pacman_anim_t;



typedef struct {
    point_t pos;
    pacman_anim_t anim;
} pacman_t;









typedef struct {
    pacman_t pacman;
} game_state_t;









void draw_sprite_from_atlas(
    uint16_t src_x,
    uint16_t src_y,
    uint16_t w,
    uint16_t h,
    uint16_t dst_x,
    uint16_t dst_y
) {


    for(uint16_t y = 0; y < h; y++){
        for(uint16_t x = 0; x < w; x++){
            uint32_t src_idx =
                (src_y+y)*Pacman_Sprite_Map__w +
                (src_x+x);
            uint32_t dst_idx =
                (dst_y+y)*SCREEN_RGB333_W +
                (dst_x+x);
            uint16_t pixel = Pacman_Sprite_Map__p[src_idx];
            unpack_rgb333_p32[dst_idx] = pixel;
        }
    }
```

стања у наредна. Ово се одиграва у другом кораку алгоритма игре, при рачунању следећег стања игре. Ово се извршава једанпут у току трајања једног фрејма, тј. 60 пута у секунди. У следећој структури чува се тренутно стање као и бројач задршке:

```
83 typedef struct {
84     pacman_anim_states_t state;
85     uint8_t delay_cnt;
86 } pacman_anim_t;
```

Бројач задршке је потребан јер ако би се директно прелазило из стања у стање, анимација би била пребрза. Овако се овај бројач постави на `PACMAN_ANIM_DELAY` и када дође до 0, прелази се на следеће стање. Наравно, могуће је направити уместо коришћења бројача задршке, још стања између горенаведених, али би то било веома заметно. Такође, на овај начин је далеко лакше подешавати брзину анимације:

```
60 #define PACMAN_ANIM_DELAY 3
```

Ако се ова повећа задршка повећа на мало већу вредност (рецимо 60 за 1 секунду), могуће је видети изгледе спрајтова који се користе у анимацији. Пошто се редовно спрајтови довољно брзо мењају, анимација ће изгледати континуално.

Даље се горепоменута структура за анимацију Pacman-а пакује у структуру Pacman заједно са његовом позицијом:

```
90 typedef struct {
91     point_t pos;
92     pacman_anim_t anim;
93 } pacman_t;
```

Да се структура даље пакује у структура стања игре, где би дошле структуре и осталих играча, односно ботова-противника, као и осталих објеката у игри.

Атлас

У горепоменутом програму се користи атлас спрајтова односно мапа спрајтова. Овакав приступ може олакшати развој игрице. Ако се погледа слика `Pacman_Sprite_Map.png` од које је изгенерисан атлас, у горњем левом углу се могу видети три спрајта која су коришћена у анимацији.

Спрајтови из ове мапе се исцртавају путем функције која за параметре узима изворишне позиције спрајта у атласу, ширине и величине спрајта, као и одредишне позиције на екрану тј. баферу:

```
110 void draw_sprite_from_atlas(
111     uint16_t src_x,
112     uint16_t src_y,
113     uint16_t w,
114     uint16_t h,
115     uint16_t dst_x,
116     uint16_t dst_y
117 ) {
```

Овде не постоји низ пиксела као параметар, јер постоји само један спрајт односно атлас, чији је низ се користи у функцији директно. Такође треба приметити да се подешавањем ширине и висине може исцртати и мањи део спрајта.

Овај пут је аритметика мало комплекснија, јер је потребно израчунати релативне индексе за изворишни и одредишни индекс:

```
120     for(uint16_t y = 0; y < h; y++){
121         for(uint16_t x = 0; x < w; x++){
122             uint32_t src_idx =
123                 (src_y+y)*Pacman_Sprite_Map__w +
124                 (src_x+x);
125             uint32_t dst_idx =
126                 (dst_y+y)*SCREEN_RGB333_W +
127                 (dst_x+x);
128             uint16_t pixel = Pacman_Sprite_Map__p[src_idx];
129             unpack_rgb333_p32[dst_idx] = pixel;
130         }
131     }
```

Спрајтови се исцртавају у фази цртања графике, где се на основу стања анимације исцртава одговарајући спрајт:

```
250         // Draw pacman.
251         switch(gs.pacman.anim.state){
252         case PACMAN_IDLE:
253         case PACMAN_OPENING_MOUTH:
254         case PACMAN_CLOSING_MOUTH:
255             // Half open mouth.
256             draw_sprite_from_atlas(
257                 16, 0, 16, 16, gs.pacman.pos.x, gs.pacman.pos.y
258             );
259             break;
260         case PACMAN_WITH_OPEN_MOUTH:
261             // Full open mouth.
262             draw_sprite_from_atlas(
263                 0, 0, 16, 16, gs.pacman.pos.x, gs.pacman.pos.y
264             );
265             break;
266         case PACMAN_WITH_CLOSED_MOUTH:
267             // Close mouth.
268             draw_sprite_from_atlas(
269                 32, 0, 16, 16, gs.pacman.pos.x, gs.pacman.pos.y
270             );
271             break;
272         }
```

Треба приметити да у неким стањима је коришћен исти спрајт. Такође треба приметити да су осим изворишне позиције спрајта, сви остали параметри исти, укључујући бетонирану ширину и висину спрајта, као и одредишне позиције преузете из структуре стања игрице.

Задатак

С овим можете већ да правите игрице, па се стога поиграјте правећи игрице. А кад направите игрицу, онда ћете моћи се поиграти с истом том игрицом. Ово вам је основа за пројекат који ће бити играца.