ЛПРС2 Лаб

Увод у графику

верзија 1.2

21. март 2021.

Покретање

Пројекат скинути са:
`https://github.com/LPRS2/LPRS2_MAX1000_Game_Console_Emulator`
Кад скинете пројекат, отворите конзолицу у `LPRS2_MAX1000_Game_Console_Emulator` фасцикли. У Ubuntu-у, може се користити Gnome-Teminal, који се може отворити менију на десни клик у горепоменутој фасцикли. У случају да то није подржано наћи га у Dash-у или отворити преко `Ctrl+Alt+T`, па променити путању до распаковане фасцикле са:

```
cd $HOME/Desktop/LPRS2_MAX1000_Game_Console_Emulator/
```

У случају Window-a, у којем се са пројектом ради преко MSYS2, отворити MSYS2 32-битну конзолицу па променити путању до распаковане фасцикле:

```
cd $USERPROFILE/Desktop/LPRS2_MAX1000_Game_Console_Emulator/
```

Пројекат користи WAF систем за компајлирање, које опет користи Python. Стога, би требао да ради на свим могућим платформама. Да ли ради видеће се кад се покрене кад се покрене прва команда испод. Сигурно ради на Ubuntu.

Када први пут покрећете овај пројекат на неком рачунару потребно покренути команду под Linux-ом (Ubuntu, Raspbian):

```
sudo ./waf prerequisites
```

Тражиће вам шифру за `sudo` права и преко `apt-get` команде ће инсталирати потребне пакете. Под Windows-ом (MSYS2) треба извршити:

```
./waf prerequisites
```

и `pacman` ће инсталирати потребне пакете. Инсталираће SFML библиотеку, неопходну за рад емулатора [1].

Када први пут покрећете отпакован или потпуно очишћен пројекат, потребно га је исконфигурисати са:

```
./waf configure
```

Ова команда ће пронаћи компајлер и библиотеке неопходне за компајлирање пројекта. У случају Windows-a, потребно је да WAF пронађе GCC и G++ компајлер од MinGW-a из MSYS2 окружења. У случају да је Visual Studio инсталиран, WAF може да повуче

[1] Иначе, SFML је одличан engine за прављење 2D игрица и тако тога. Топла препорука да га користите.

MSVC као компајлер, што неће радити после у компајлирању. У том случају покренути конфигурацију са додатним опцијама:

```
./waf configure --check-c-compiler=gcc --check-cxx-compiler=g++
```

Ове опције ће натерати WAF да користи MinGW компајлере из MSYS2 окружења.

Могуће је заобићи WAF и радити са Code Blocks или Visual Studio окружењем. На интернету постоје туторијали за SFML за оба окружења. Ако овде постоје неки проблеми гађајте Гугл или асистенте.

Након тога се пројекат компајлира са:

```
./waf build
```

Сваки пут када се промени код, потребно је наново искомпајлирати пројекат. Програм покренути са:

```
./waf run --app=intro
```

Ради аутоматизованијег рада, ове две команде се могу саставити овако:

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

У случају да постоје некакви проблеми при раду програма, поготово перформанса, могуће је подесити емулатор преко CONFIG.H заглавља.

За чишћење само бинарних фајлова, резултата компајлирања, покренути:

```
./waf clean
```

За потпуно чишћење пројекта урадити:

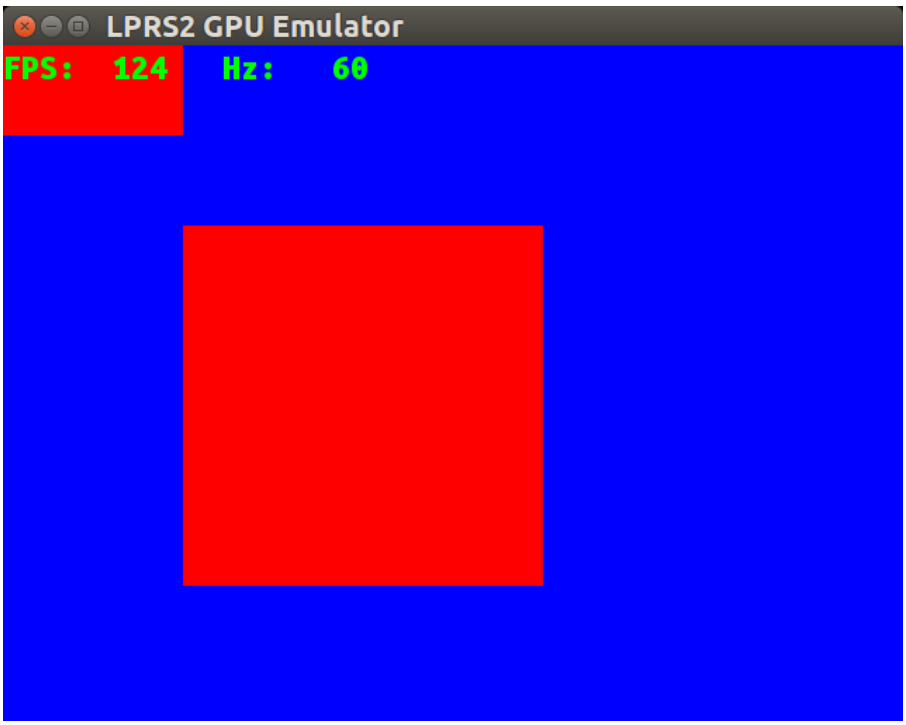
```
./waf distclean
```

Након тога могуће је пројекат запаковати са:

```
./waf dist
```

Изглед

Када се покрене програм отвориће се прозор са следећим изгледом:



Слика 1: Почетни изглед екрана емулатора

На екрану су приказана плава позадина; црвени правоугаоник и квадрат, који представљају играче; и зелени заслон (енгл. HUD - Head Up Display). Заслон приказује број фрејмова у секунди (енгл. FPS - Frames Per Second) односно колико графичка може да исцрта фрејмова у секунди; такође се приказује и фреквенцију приказивања. Почетно подешавање за њу је 60 Hz. Ако пада испод те вредности, значи да је графичка преспора за фреквенцију приказивања, свакако је могуће спустити фреквенцију приказивања у CONFIG.H и до 25 Hz.

У једном тренутку је само један играч активан. Активан играч се покреће на стрлице тастатуре. Са тастером **A** се мења активан играч. Из програма емулатора се може изаћи преко **Esc** тастера.

Графичка

Графичка (енгл. GPU - Graphical Procesing Unit) има следеће модове:

Табела 1: Модови графичке

Мод	Код	Резолуција	Број боја
Color bar мод	0	NA	NA
IDX1 (тј. *1b index*) мод	1	640x480	2
IDX4 (тј. *4b index*) мод	2	320x240	16
RGB333 (тј. *9b color*) мод	3	160x120	512

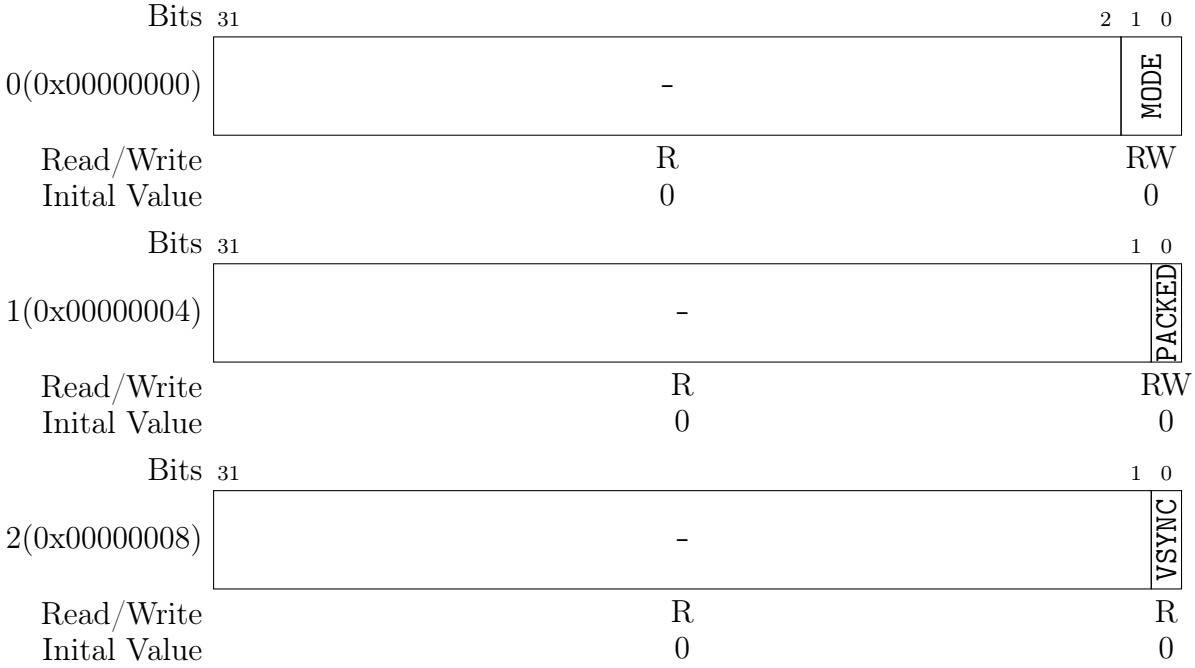
Обојени ступци (енгл. Color bar) мод служи за тестирање да ли графички хардвер ради, и функционише независно да ли је нешто нацртано из апликације. Примера ради, ако нема ништа нацртано на екрану, погодно је пребацити на овај мод, како би се тестирао хардвер, веза са монитором, монитор, итд. Ако то ради у Color bar моду, хардвер је елиминисан као кривац, и проблем је у апликацији. Остали модови су познати са предавања, па овде неће бити даље разглабано о њима.

Меморијска мапа графичке има 4 дела:

Табела 2: Региони мем. мапе графичке

Регион	Почетна адреса
Common	`0x000000`
IDX1 mode	`0x400000`
IDX4 mode	`0x800000`
RGB333 mode	`0xc00000`

У *Common* региону се налазе 3 основна регистра:



Слика 2: Основни регистри

У `MODE` регистар се уписује код активног мода. У `PACKED` регистар се уписује 0 да буде активан распаковане, односно 1 за паковане бафере. Регистар `VSYNC` се полује како би се добио статус вертикалног синхронизационог сигнала, ради синхронизације апликационог кода тј. игре. Вертикална синхронизација је период у исцртавању слике на екрану између две слике (енгл. Frame). Овај период потиче из доба екрана са кадоним цевима, када је било потребно одређено време да се сноп електрона, након што је исцртао слику на екрану линију по линију, врати из доњег десног на горњи леви угао, како би могао цртати нову слику.

У *Common* региону се такође налази палета за индексне модове. Палета почиње на адреси `0x1000`. Сваки 32-битни регистар је једна боја у палети у RGB888 формату. Ако је активан *IDX1* мод, користиће се прва 2 регистра из палете, `0x1000` и `0x1004`; У *IDX4* се користи јелте 16 комада, од 0. до 15. регистра. Овде треба напоменути, да се на емулатору на адреси `0x2000` боја за заслон (енгл. HUD - Head Up Display). Боја заслона се може по потреби променити, ако се поклапа са садржајем који се исцртава на екрану, превасходно позадином.

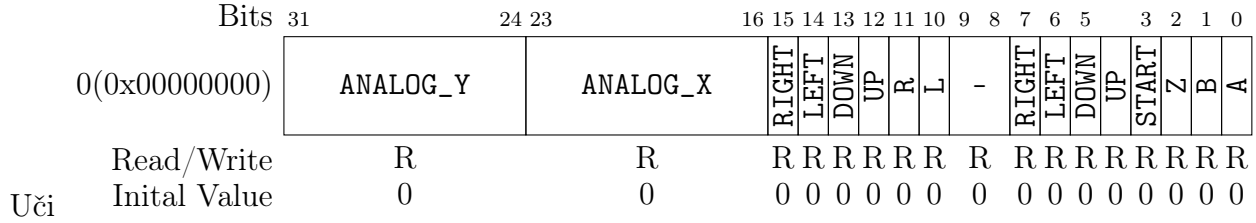
У остала 3 региона налазе се бафери истоимених модова.

Табела 3: Бафери

Регион	Паковање	Адреса
IDX1 mode	unpacked packed	0x400000 0x600000
IDX4 mode	unpacked packed	0x800000 0xa00000
RGB333 mode	unpacked	0xc00000

Џојпед

Џојпед (енгл. Joypad) има само један регистар са следећим контролама:



Учи

Слика 3: Регистри Џојпеда

На емулатору `ANALOG_X` и `ANALOG_Y` нису имплементирани.

Опис

Заронимо сада у код. На следећим линијама могу се видети сада већ познате дефиниције показивача.

```
18 #define gpu_p32 ((volatile uint32_t*)LPRS2_GPU_BASE)
19 #define palette_p32 ((volatile uint32_t*)(LPRS2_GPU_BASE+0x1000)
   )
20 #define unpack_idx1_p32 ((volatile uint32_t*)(LPRS2_GPU_BASE+0
   x400000))
21 #define pack_idx1_p32 ((volatile uint32_t*)(LPRS2_GPU_BASE+0
   x600000))
22 #define joypad_p32 ((volatile uint32_t*)LPRS2_JOYPAD_BASE)
23
24 typedef struct {
25     unsigned a      : 1;
26     unsigned b      : 1;
27     unsigned z      : 1;
28     unsigned start  : 1;
29     unsigned up     : 1;
30     unsigned down   : 1;
31     unsigned left   : 1;
32     unsigned right  : 1;
33 } bf_joypad;
34 #define joypad (*((volatile bf_joypad*)LPRS2_JOYPAD_BASE))
```

Овде се уместо дефинисања промењливих су дефинисани макрои. Интересантна је последња линија у листингу, где се показивач одмах и дереференцира тако да се не мора користити оператор `->`.

Следећи излист приказује конфигурацију игрице.

```
46 #define STEP 32
47 #define RECT_H 64
48 #define RECT_W 128
49 #define SQ_A 256
50
51 #define UNPACKED_0_PACKED_1 0
```

Од значаја је `UNPACKED_0_PACKED_1`, с којим се мења да ли се користе паковани или распаковани бафери.

Следећи излист приказује структуре за опис стања игре (енгл. Gamestate)

```
58 typedef struct {
59     uint16_t x;
60     uint16_t y;
61 } point_t;
62
63 typedef enum {RECT, SQ} player_t;
64
65 typedef struct {
66     // Upper left corners.
67     point_t rect;
68     point_t sq;
69
70     player_t active;
71 } game_state_t;
```

Као што је било речено на предавањима, алгоритам игре почиње са **подешавањима и иницијализацијама** и након тога се у петљи изврашавају кораци:

- **Учитавање контрола,**
- **Рачунање следећег стања,**
- **Исцртавање.**

Прво се конфигурише графичка.

```
80     // Setup.
81     gpu_p32[0] = 1; // 1b index mode.
82     gpu_p32[1] = UNPACKED_0_PACKED_1;
83     palette_p32[0] = 0x00ff0000; // Blue for background.
84     palette_p32[1] = 0x000000ff; // Red for players.
```

Након тога се иницијализује стање игрице.

```
88      // Game state.
89      game_state_t gs;
90      gs.rect.x = 0;
91      gs.rect.y = 0;
92
93      gs.sq.x = 128;
94      gs.sq.y = 128;
95
96      gs.active = RECT;
```

У петљи се прво врши учитавање контрола.

```
104         // Poll controls.
105         int mov_x = 0;
106         int mov_y = 0;
107         if(joypad.down){
108             mov_y = +1;
109         }
110         if(joypad.up){
111             mov_y = -1;
112         }
113         if(joypad.right){
114             mov_x = +1;
115         }
116         if(joypad.left){
117             mov_x = -1;
118         }
119         //TODO Have bug here. Hold right button and play with A
    button.
120         int toggle_active = joypad.a;
```

Резултат се складишти у међу-промењиве.
Након тога се врши рачунање следећег стања игре.

```
130         switch(gs.active){
131         case RECT:
132             //TODO Limit not to go through wall. Same for all
    players.
133             gs.rect.x += mov_x*STEP;
134             gs.rect.y += mov_y*STEP;
135             if(toggle_active){
136                 gs.active = SQ;
137             }
138             break;
139         case SQ:
140             gs.sq.x += mov_x*STEP;
141             gs.sq.y += mov_y*STEP;
142             if(toggle_active){
143                 gs.active = RECT;
144             }
145             break;
146         }
```

Овде треба запазити како на сличан начин у VHDL-у се пише `switch` по неком стању и ту на основу улаза и тренутног стања рачуна наредно стање.

Након тога се врши исцртавање. Први корак јесте извршити синхронизацију програма и графичке. Наиме, пошто се овде користи само један бафер оквира, а не двострико баферовање, потребно је цртати када је графичка у стању вертикалне синхронизације.

```
154         // Detecting rising edge of VSync.
155         WAIT_UNITL_0(gpu_p32[2]);
156         WAIT_UNITL_1(gpu_p32[2]);
157         // Draw in buffer while it is in VSync.
```

Синхронизација се обавља заузетим чекањем (енгл. Busy wait) растуће ивице сигнала вертикалне синхронизације прозивком `VSYNC` регистар. Након тога је графичка у вертикалној синхронизацији, и могуће је цртати у бафер оквира без артифакта на екрану.

Прво се исцрата плава позадина.

```
164         // Clear to blue.
165         for(int r = 0; r < SCREEN_H; r++){
166             for(int c = 0; c < SCREEN_W; c++){
167                 unpack_idx1_p32[r*SCREEN_W + c] = 0;
168             }
169         }
```

У локације регистра се уписује `0`, која индексира плаву боју из палете. Користи се распаковани бафер 1-битног индексног мода графичке. Пошто је показивач на тај бафер 1-димензионални низ, потребно је срачунати индексирати тај 1-димензионални низ као 2-димензионални. То се обавља изразом `r*SCREEN_W + c`. Пошто су у C-у редови у предности (енгл. Row Major), онда ће локације колоне `c` бити узастопне, док ће редови почињати на сваких `SCREEN_W` локација. Пример ради, ако би имали матрицу са 2 реда и 3 колоне, изгледала би овако:

$$M = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

Нека је притом `M` матрица, `A` низ, а `W` ширина матрице, у овом случају 3. Она би код за индексирање свагок елемента изгледао као у табели испод.

Табела 4: Индексирање матрице у C-у

Ред	Колона	Преко матрице	Преко низа	Адреса	Вредност
0	0	`M[0][0]`	`A[0*W + 0]`	0	0
0	1	`M[0][1]`	`A[0*W + 1]`	1	1
0	2	`M[0][2]`	`A[0*W + 2]`	2	2
1	0	`M[1][0]`	`A[1*W + 0]`	3	3
1	1	`M[1][1]`	`A[1*W + 1]`	4	4
1	2	`M[1][2]`	`A[1*W + 2]`	5	5

Треба запазити да је спољашња метља по редовима, а унутрашња по колонама.

```
165         for(int r = 0; r < SCREEN_H; r++){
166             for(int c = 0; c < SCREEN_W; c++){
```

Разлог за то је да се чешће приступа узастопним локацијама. На тај начин се користи просторна локалност (енгл. Spacial Locality) скривене меморије (енгл. Cache).

На сличан начин се исцртава и правоугаоник.

```
175         for(int r = gs.rect.y; r < gs.rect.y+RECT_H; r++){
176             for(int c = gs.rect.x; c < gs.rect.x+RECT_W; c++){
177                 unpack_idx1_p32[r*SCREEN_W + c] = 1;
178             }
179         }
```

Притом се користе макрои из конфигурације игре и чланови из стуктуре стања игре за срачунавање граница правоугаоник тј. за почетне и крајње индексе редова и колона. У регистар се уписује 1, како би се индексирала црвена боја из палете.

Приликом цртања квадрата се користи један 2-димензионалан приступ. У C-у није могуће дефинисати показивач на 2-димензионални низ, у који се може доделити нека адреса, а самим ти није могуће конвертовати адресу у показивач 2-димензионалног низа. Решење за то је направити структуру са дефиницијом 2-димензионалног низа, па показивачу на ту структуру доделити као на излисту испод:

```
36 typedef struct {
37     uint32_t m[SCREEN_H][SCREEN_W];
38 } bf_unpack_idx1;
39 #define unpack_idx1 (*((volatile bf_unpack_idx1*)unpack_idx1_p32
      ))
```

После тога се лако може обавити 2-димензионално индексирање бафера.

```
183         // Red sqaure.
184         // Use struct with 2D matrix.
185         for(int r = gs.sq.y; r < gs.sq.y+SQ_A; r++){
186             for(int c = gs.sq.x; c < gs.sq.x+SQ_A; c++){
187                 unpack_idx1.m[r][c] = 1;
188             }
189         }
```

У свим горенаведеним примерима цртања користе се распаковани бафери, где је у сваки 32-битни регистар спакован 1 1-битни пиксел. Иако је овакав начин приступа лакши за цртkање, изузетно је спор. Из тог разлога погодно је користити паковане бафере, где је у један 32-битни регистар спаковано 32 пиксела. Самим тим је овај приступ 32 пута бржи. Излист испод приказује овакав приступ:

```
198         for(int r = 0; r < 32; r++){
199             for(int c = 0; c < 1; c++){
200                 pack_idx1_p32[r*(SCREEN_W/32) + c] = 0xffffffff;
201             }
202         }
```

Основна разлика је у ширини тј. броју колона, који је сад 32 пута мањи. Незгодација са оваквим приступом је када су потребни облици чија ширина није умножак 32 колоне. У том случају је потребно срачунати вредност која треба да се упише у регистар. Ради тестирања овог начина цртања потребно је макро `UNPACKED_0_PACKED_1` поставити на `1`.

Задатак

Поиграти се мало са игрицом. Издебаговати баг са мењањем активног играча. Реализовати ограничење да играч не пролази кроз зид. Реализовати игру да функционише са пакованим бафером. Додати троугао и круг (у оба мода).