Projekt

Sterowniki robotów

Dokumentacja

Rozbudowana wersja Prymitywnego Animowanego Wyświetlacza w języku C

PAW 3.14

Skład grupy:
Jakub Cebulski, 235773
Miłosz Chlebowski, 235427

Termin: srTP15

Prowadzący:
mgr inż. Wojciech Domski

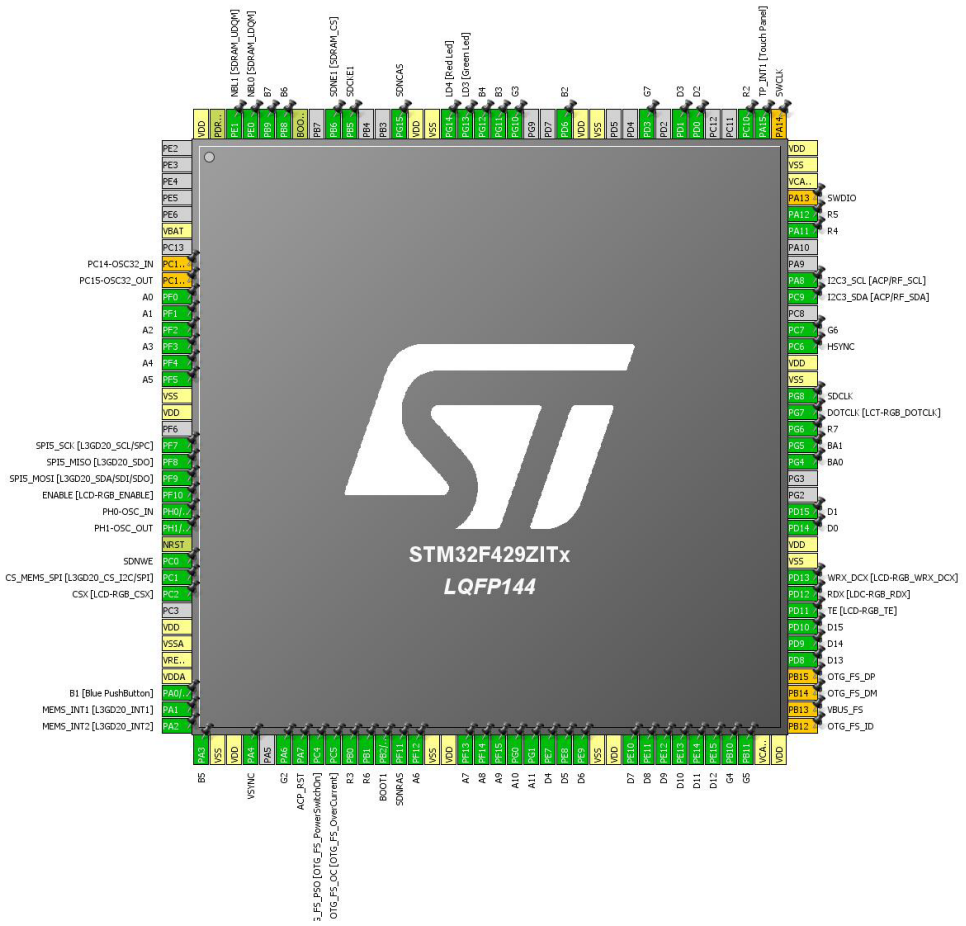
25 czerwca 2019

Spis treści

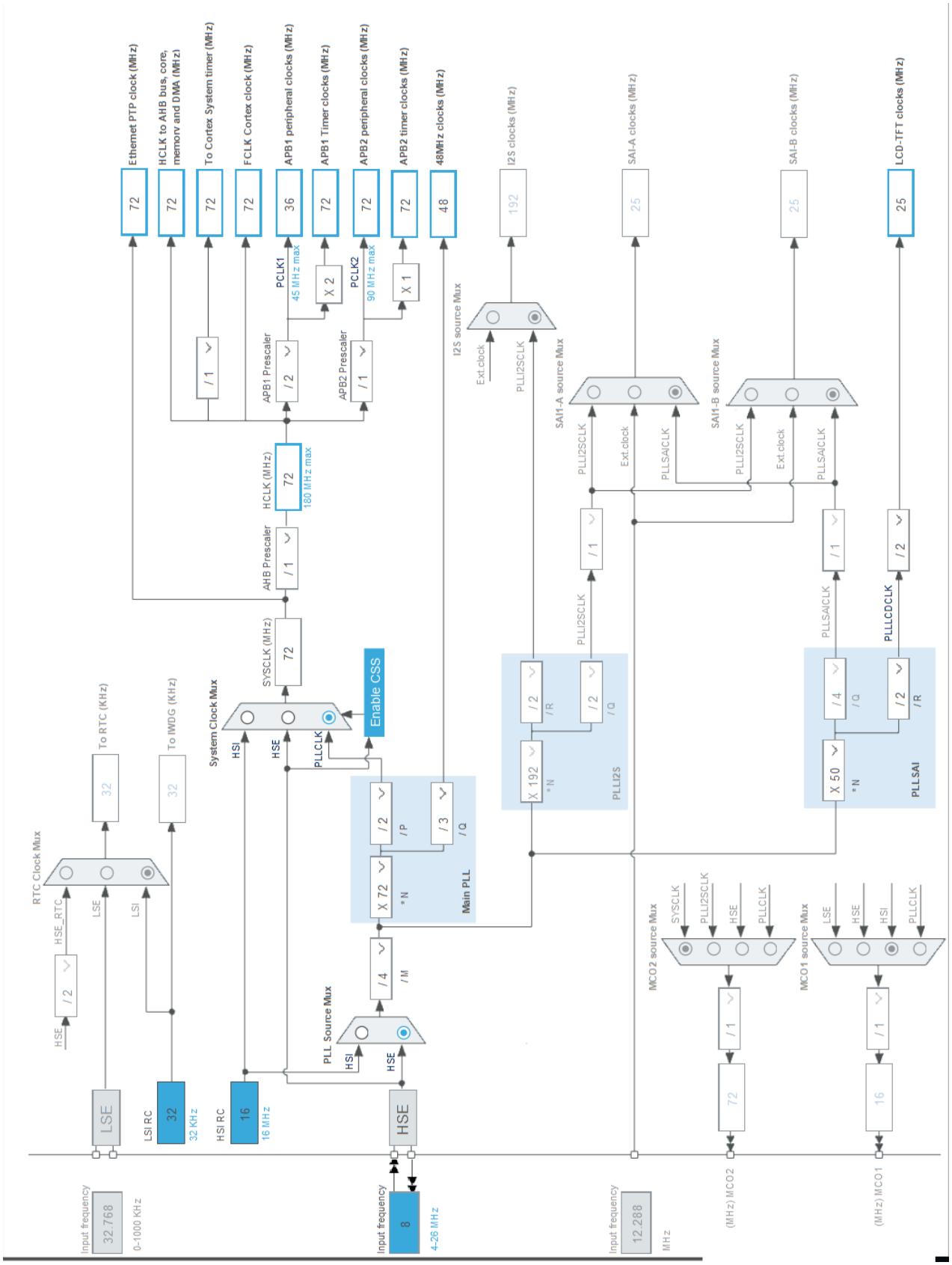
1 Opis projektu

Projekt jest kontynuacją projektu PAW, jest to otwartoźródłowa biblioteka graficzna dla mikrokontrolerów STM32, stworzona i rozwijana w ramach projektów studenckich w poprzednich latach.

2 Konfiguracja mikrokontrolera



Rysunek 1: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



Rysunek 2: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
8	PC14/OSC32_IN*	RCC_OSC32_IN	PC14-OSC32_IN
9	PC15/OSC32_OUT*	RCC_OSC32_OUT	PC15-OSC32_OUT
10	PF0	FMC_A0	A0
11	PF1	FMC_A1	A1
12	PF2	FMC_A2	A2
13	PF3	FMC_A3	A3
14	PF4	FMC_A4	A4
15	PF5	FMC_A5	A5
19	PF7	SPI5_SCK	SPI5_SCK [L3GD20_SCL/SPC]
20	PF8	SPI5_MISO	SPI5_MISO [L3GD20_SDO]
21	PF9	SPI5_MOSI	SPI5_MOSI [L3GD20_SDA/SDI/SDO]
22	PF10	LTDC_DE	ENABLE[LCD-RGB_ENABLE]
23	PH0/OSC_IN	RCC_OSC_IN	PH0-OSC_IN
24	PH1/OSC_OUT	RCC_OSC_OUT	PH1-OSC_OUT
26	PC0	FMC_SDNWE	SDNWE
27	PC1	GPIO_Output	NCS_MEMS_SPI[L3GD20_CS_I2C/SPI]
28	PC2	GPIO_Output	CSX [LCD-RGB_CSX]
34	PA0/WKUP	GPIO_EXTI0	B1 [Blue PushButton]
35	PA1	GPIO_EXTI1	MEMS_INT1 [L3GD20_INT1]
36	PA2	GPIO_EXTI2	MEMS_INT2 [L3GD20_INT2]
37	PA3	LTDC_B5	B5
40	PA4	LTDC_VSYNC	VSYNC
42	PA6	LTDC_G2	G2
43	PA7	GPIO_Output	ACP_RST
44	PC4	GPIO_Output	OTG_FS_PSO [OTG_FS_PowerSwitchOn]
45	PC5	GPIO_EXTI5	OTG_FS_OC [OTG_FS_OverCurrent]
46	PB0	LTDC_R3	R3
47	PB1	LTDC_R6	R6
48	PB2/BOOT1	GPIO_Input	BOOT1
49	PF11	FMC_SDNRAS	SDNRAS
50	PF12	FMC_A6	A6
53	PF13	FMC_A7	A7
54	PF14	FMC_A8	A8
55	PF15	FMC_A9	A9
56	PG0	FMC_A10	A10
57	PG1	FMC_A11	A11
58	PE7	FMC_D4	D4
59	PE8	FMC_D5	D5
60	PE9	FMC_D6	D6
63	PE10	FMC_D7	D7
64	PE11	FMC_D8	D8
65	PE12	FMC_D9	D9
66	PE13	FMC_D10	D10
67	PE14	FMC_D11	D11
68	PE15	FMC_D12	D12
69	PB10	LTDC_G4	G4
70	PB11	LTDC_G5	G5
73	PB12*	USB_OTG_HS_ID	OTG_FS_ID
74	PB13*	USB_OTG_HS_VBUS	VBUS_FS
75	PB14*	USB_OTG_HS_DM	OTG_FS_DM
76	PB15*	USB_OTG_HS_DP	OTG_FS_DP
77	PD8	FMC_D13	D13
78	PD9	FMC_D14	D14
79	PD10	FMC_D15	D15
80	PD11	GPIO_Input	TE [LCD-RGB_TE]
81	PD12	GPIO_Output	RDX [LDC-RGB_RDX]
82	PD13	GPIO_Output	WRX_DCX [LCD-RGB_WRX_DCX]
85	PD14	FMC_D0	D0
86	PD15	FMC_D1	D1
89	PG4	FMC_BA0	BA0

90	PG5	FMC_BA1	BA1
91	PG6	LTDC_R7	R7
92	PG7	LTDC_CLK	DOTCLK [LCT-RGB_DOTCLK]
93	PG8	FMC_SDCLK	SDCLK
96	PC6	LTDC_HSYNC	HSYNC
97	PC7	LTDC_G6	G6
99	PC9	I2C3_SDA	I2C3_SDA [ACP/RF_SDA]
100	PA8	I2C3_SCL	I2C3_SCL [ACP/RF_SCL]
103	PA11	LTDC_R4	R4
104	PA12	LTDC_R5	R5
105	PA13*	SYS_JTMS-SWDIO	SWDIO
109	PA14*	SYS_JTCK-SWCLK	SWCLK
110	PA15	GPIO_EXTI15	TP_INT1 [Touch Panel]
111	PC10	LTDC_R2	R2
114	PD0	FMC_D2	D2
115	PD1	FMC_D3	D3
117	PD3	LTDC_G7	G7
122	PD6	LTDC_B2	B2
125	PG10	LTDC_G3	G3
126	PG11	LTDC_B3	B3
127	PG12	LTDC_B4	B4
128	PG13	GPIO_Output	LD3 [Green Led]
129	PG14	GPIO_Output	LD4 [Red Led]
132	PG15	FMC_SDNCAS	SDNCAS
135	PB5	FMC_SDCKE1	SDCKE1
136	PB6	FMC_SDNE1	SDNE1 [SDRAM_CS]
139	PB8	LTDC_B6	B6
140	PB9	LTDC_B7	B7
141	PE0	FMC_NBL0	NBL0 [SDRAM_LDQM]
142	PE1	FMC_NBL1	NBL1 [SDRAM_UDQM]

Tabela 1: Konfiguracja pinów mikrokontrolera

2.2 DMA2D

DMA2D mode and configuration	
Basic Parameters:	
Transfer Mode	Memory to memory
Color Mode	RGB565
Output Offset	0
Foreground layer Configuration:	
DMA2D Input Color Mode	RGB565
DMA2D ALPHA MODE	No modification of the alpha channel value

2.3 LTDC

LTDC mode and configuration	
Display Type	RGB565 (16bits)
Synchronization for Width	
Horizontal Synchronization Width	8
Horizontal Back Porch	7
Active Width	320
Horizontal Front Porch	6
HSync Width	7
Accumulated Horizontal Back Porch Width	14
Accumulated Active Width	334
Total Width	340
Synchronization for Height	
Vertical Synchronization Height	4
Vertical Back Porch	2
Active Height	480
Vertical Front Porch	2
VSync Height	3
Accumulated Vertical Back Porch Height	5
Accumulated Active Height	485
Total Height	487
Signal Polarity	
Horizontal Synchronization Polarity	Active Low
Vertical Synchronization Polarity	Active Low
Data Enable Polarity	Active Low
Pixel Clock Polarity	Normal Input
BackGround Color	
Red	0
Green	0
Blue	0

3 Opis działania programu

Przyjęte konwencje

Program napisano zgodnie z paradygmatem programowania obiektowego. W związku z tym tworzone struktury posiadały swoje metody oznaczone nazwą struktury jako przedrostkiem nazwy funkcji. Ponadto niektóre złożone struktory posiadają konstruktor i destruktor. W związku z brakiem bezpośredniego wsparcia w języku C były to zwykłe funkcje.

3.1 Opis działania podwójnego buforowania

Podwójne buforowanie zostało zrealizowane dzięki podmianie adresu bufora po każdym zakończonym rysowaniu klatki. Na początku funkcji `PAW_Scene_display`, ustalany jest adres aktualnego bufora. Po narysowaniu klatki do pamięci, DMA2D jest informowane o zmianie adresu, z którego ma przesyłać dane do wyświetlacza. Podwójny bufor można wyłączyć zmieniając działanie funkcji `PAW_Buffer_Switch` tak aby zwracała za każdym razem ten sam adres pamięci SDRAM.

3.2 Teksturowanie

Biblioteka umożliwia nakładanie nakładanie tekstur na kwadraty. Tekstury są tworzone poprzez umieszczanie figur 2D na określonym obszarze. Umieszczając kwadraty z teksturami w przestrzeni 3D pozwala tworzyć barwne bryły. Dzięki sortowaniu kwadratów przed wyświetleniem otrzymano realistyczny efekt przesłaniania się ich nawzajem.

3.3 Biblioteka matematyczna

Biblioteka matematyczna składa się z trzech modułów. Moduł PAW_Math zawiera stałe i funkcje wykorzystywane przez resztę biblioteki. PAW_Vector jest reprezentają wektora o arbitralnej liczbie wy-

miarów. Pozwala na wykonywanie na nich operacji matematycznych takich jak dodawanie i odejmowanie ich od siebie a także mnożenie i dzielenie przez skalary. PAW_Matrix pozwala na operacje na macierzach kwadratowych o dowolnej wielkości, między innymi liczenie odwrotności, dodawanie ich do siebie oraz mnożenie przez wektor lub skalar.

3.4 Opis działania aplikacji pokazowej

Aplikacja testowa składa się z czterech przykładowych zastosowań biblioteki. Można je przełączać używając przycisku na płytce.

Pierwsze z nich to prosta animacja w 2D. Pokazuje płynność obrazu osiągniętą dzięki podwójnemu buforowi.

Drugie to kostka będąca elementem projektu PAW-2.0. Pokazuje ono udaną reimplementację grafiki 3D w języku C.

Kolejna widoczna kostka składa się już z nowo utowrzonych struktur - kwadratów oraz posiada nałożone tekstury. Dzięki sortowaniu obiektów od najdalszych do najbliższych osiągnięto efekt przysłaniania się ścian. Animacja nie jest płynna ze względu na to że do każdej klatki kostka jest generowana od nowa wraz z teksturami.

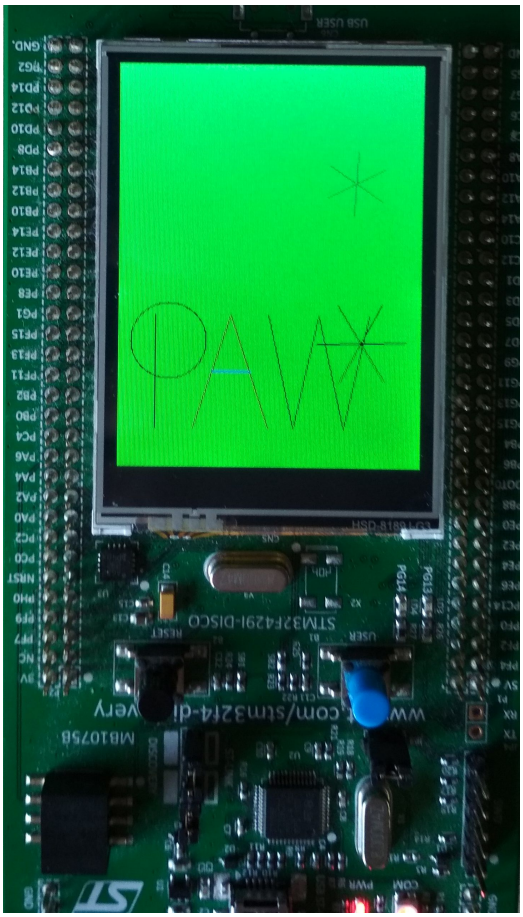
Ostatnie demo prezentuje to samo co poprzednie, tym razem z użyciem projekcji 3D do 2D do nowej figury. Dzięki temu zamiast tworzyć nowe kostki obracana jest wciąż ta sama a płynność animacji jest znacznie większa. Jest to funkcjonalność wykraczająca poza założenia projektu, ze względu na ograniczenia czasowe nie została dopracowana i powoduje wycieki pamięci.

Tworzenie i wyświetlanie kształtów w 2D

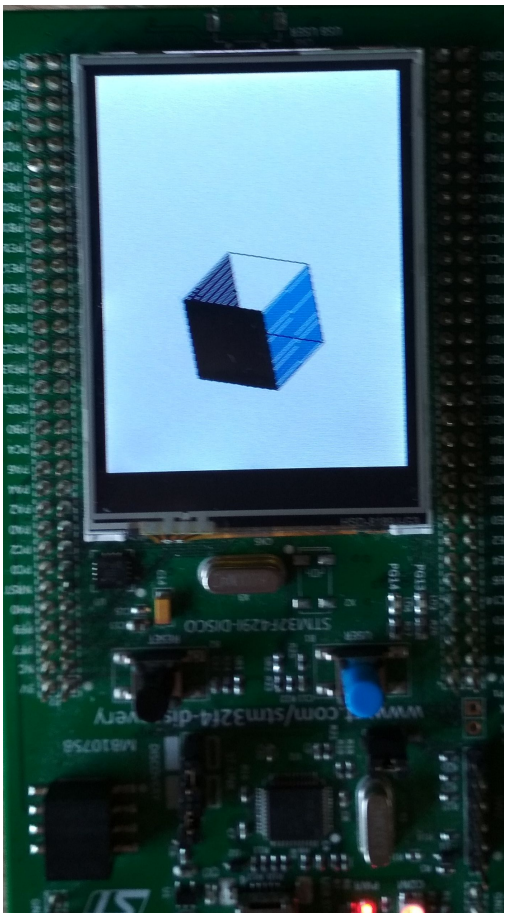
Biblioteka PAW pozwala na tworzenie kształtów takich jak okręgi czy odcinki wykorzystując wektory zaimplementowane w bibliotece matematycznej do określenia pozycji w trójwymiarowym układzie współrzędnych. Macierze pozwalają na dokonywanie translacji oraz rotacji.

Tworzenie i wyświetlanie kształtów w 3D

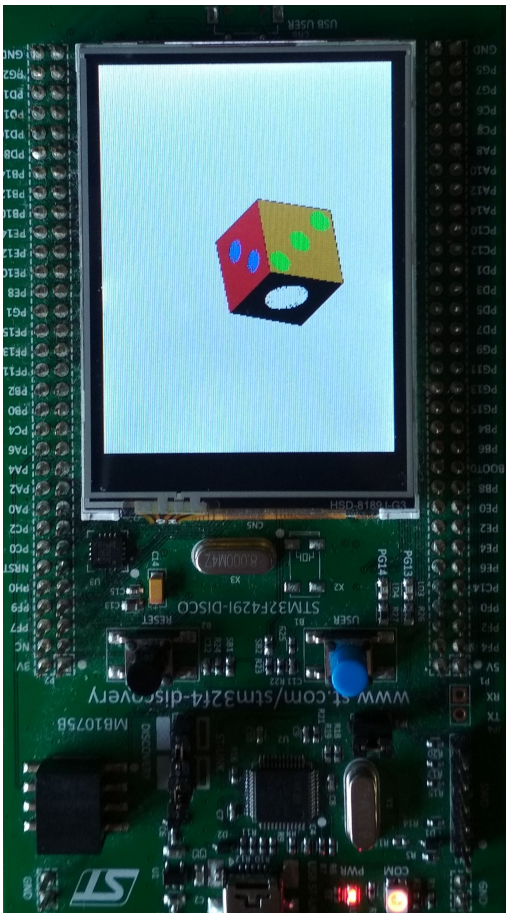
Poza figurami w 2D biblioteka pozwala na tworzenie brył. Składają się one z tych samych prymitywów co figury 2D, jednak posiadają własne metody translacji i rotacji. Dodatkowo pozwalają na projekcję do 2D przy pomocy jednej z 2 metod. Pierwsza z nich dokonuje tej operacji w miejscu, druga tworzy nową figurę.



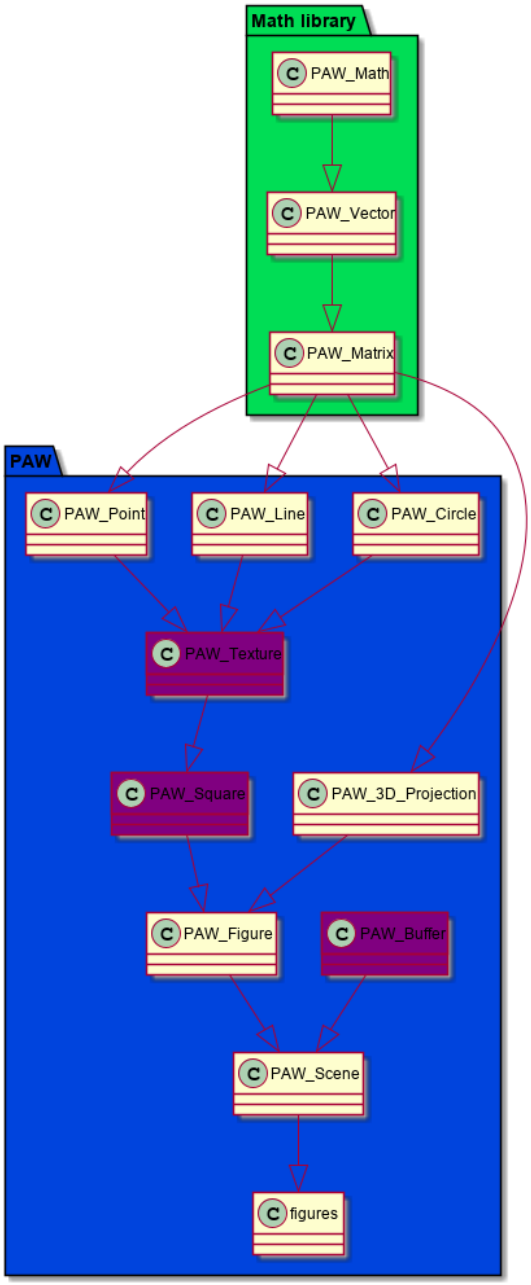
Rysunek 3: Demo 2D



Rysunek 4: Demo 3D



Rysunek 5: Demo 3D z teksturami



Rysunek 6: Diagram modułów

4 Zarządzanie projektem

W celu organizacji pracy nad kodem zastosowano system kontroli wersji git. Repozytorium z kodem znajduje się pod adresem `https://gitlab.com/mchlebowski/paw-3.14`.

Poszczególne zadania realizowano na przeznaczonych do tego gałęziach. Następnie po zaakceptowaniu zmian przez wszystkich członków grupy scalano je z gałęzią develop. W gałęzi master znalazł się kod ukończonego projektu.

5 Podsumowanie

Mimo trudności, osiągnięto wszystki postawione cele. Szczególne problemy sprawiła implementacja podwójnego buforowania, mimo że finalne rozwiązanie jest trywialne. Użyte narzędzia w postaci języka

C oraz biblioteki HAL okazały się wystarczające do realizacji założeń projektu. Wiele opracji, w szczególności zarządzanie pamięcią mogłoby się okazać znacznie prostszymi mając do dyspozycji możliwości oferowane przez język C++. Dodatkowo uprościłoby to użycie biblioteki z perspektywy użytkownika końcowego.

Literatura

[1] T. Francuz. *Mikrokontrolery AVR i ARM : sterowanie wyświetlaczami LCD*. 2017.

[2] M. A. Galewski. *STM32 : aplikacje i ćwiczenia w języku C* . 2011.