3. Zmiana napisu zależnie od stanu przycisku
Opcja 1 – czyszczenie całego ekranu (niezalecane, zajmuje więcej czasu i czyści cały ekran, ale można im pokazać żeby przekonać ich że nie każdy pomysł to dobry pomysł i czasem trzeba wpaść na coś lepszego):

```
BSP_LCD_Clear(LCD_COLOR_WHITE);
BSP_LCD_SetFont(&Font16);
BSP_LCD_SetTextColor(LCD_COLOR_BLACK);

while(1)
{
	if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
	{
		BSP_LCD_Clear(LCD_COLOR_WHITE);
		BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) "Przycisk wcisniety", CENTER_MODE);
	}
	else
	{
		BSP_LCD_Clear(LCD_COLOR_WHITE);
		BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) "Przycisk puszczony", CENTER_MODE);
	}
	HAL_Delay(50);
}
```

Opcja 2 – będziemy sprytni i zamiast czyścić cały ekran będziemy pisać dokładnie ten sam napis który chcemy skasować ale w kolorze tła – dzięki temu usuniemy dokładnie tyle pikseli ile potrzeba, oszczędzając cykle zegarowe. Poza tym zamiast co powtórzenie pętli pisać na nowo tekst będziemy to robić tylko jeśli nastąpiła zmiana stanu przycisku

```
BSP_LCD_Clear(LCD_COLOR_WHITE);
BSP_LCD_SetFont(&Font16);
BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) "Przycisk puszczony", CENTER_MODE);

int stan_poprzedni = 0;
int stan_aktualny = 0;

while(1)
{
	if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
		stan_aktualny = 1;
	else
		stan_aktualny = 0;

	if(stan_aktualny != stan_poprzedni)
	{
		if(stan_aktualny == 1)
		{
			BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
			BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) "Przycisk puszczony", CENTER_MODE);
			BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
			BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) "Przycisk wcisniety", CENTER_MODE);
		}
		else
		{
			BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
			BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) "Przycisk wcisniety", CENTER_MODE);
			BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
			BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) "Przycisk puszczony", CENTER_MODE);
		}
	}
	stan_poprzedni = stan_aktualny;
	HAL_Delay(50);
}
```

Oczywiście “przycisk wciśnięty” i “przycisk puszczony” można zmienić na dowolny inny ciąg znaków. Jeśli nie mieści się na wyświetlaczu zmniejszyć rozmiar czcionki.

4. Prosta animacja
Odbijajaca się kulka:

```
int promien = 20;
BSP_LCD_Clear(LCD_COLOR_WHITE);

while(1)
{
	for(int y = promien ; y < 320 - promien ; y++)
	{
		BSP_LCD_SetTextColor(LCD_COLOR_RED);
		BSP_LCD_FillCircle(120, y, promien);
		HAL_Delay(10);
		BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
		BSP_LCD_DrawCircle(120, y, promien);
	}

	for(int y = 320 - promien ; y > 20 ; y--)
	{
		BSP_LCD_SetTextColor(LCD_COLOR_RED);
		BSP_LCD_FillCircle(120, y, promien);
		HAL_Delay(10);
		BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
		BSP_LCD_DrawCircle(120, y, promien);
	}
}
```

Odbijanie w poziomie – wszystkie 320 (wysokość LCD) zamienić na 240 (szerokość LCD), y zamienić na x, rysowanie koła zmienić na *BSP_LCD_FillCircle(x, 160, promien);*
Dla wyjaśnienia – dlaczego rysujemy czerwone koło, ale zmazujemy (rysujemy na biało) tylko okrąg? Ano dlatego że operacja rysowania wypełnionego kształtu jest, w zależności od jego rozmiaru, kilkaset do kilkunastu tysięcy bardziej czasochłonna od rysowania pustego (wystarczy policzyć ile pikseli jest zamalowanych w pełnym kole, a ile w samym okręgu i porównać te dwie wartości), a ponieważ koło przesuwa się tylko o jeden piksel, a okrąg ma grubość dokładnie jednego piksela, wystarczy zmazać sam okrąg i nie zmieni to efektu.
Podobnie w przypadku prostokąta – rysować za pomocą funkcji FillRect, ale kasowac z użyciem DrawRect.

Jest to tylko jeden przykład animacji jaką można wykonać z użyciem kształtów, wymyślić można ich sporo więcej.

5. To samo co 4, wystarczy tylko w każdym forze po delay wstawić:
while(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
Dzięki temu pętla for zatrzyma się na tak długo jak wciśnięty jest przycisk. Można również zrobić na odwrót – animacja działa tylko przy wciśniętym przycisku, wtedy zamiast powyższego wstawić:
while(!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));

6. Wyświetlanie zmieniającej się zmiennej

```
BSP_LCD_Clear(LCD_COLOR_WHITE);
BSP_LCD_SetFont(&Font12);
BSP_LCD_SetTextColor(LCD_COLOR_BLACK);

int czas = 0;
char bufor[] = {0};

while(1)
{
	BSP_LCD_SetTextColor(LCD_COLOR_BLACK);
	sprintf(bufor, "Od resetu minelo %d sekund", czas);
	BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) bufor, CENTER_MODE);
	HAL_Delay(1000);
	BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
	BSP_LCD_DisplayStringAt(0, 10, (uint8_t*) bufor, CENTER_MODE);
	czas++;
}
```

Pomiar czasu nie będzie bardzo dokładny, ale chodzi o samo to że zmienna się zmienia i jest co jakiś czas aktualizowana na wyświetlaczu.

7. Kostka do gry

```
BSP_LCD_Clear(LCD_COLOR_WHITE);
BSP_LCD_SetFont(&Font16);
BSP_LCD_SetTextColor(LCD_COLOR_BLACK);

int wynik_rzutu = 0;
int stan_poprzedni = 0;
int stan_aktualny = 0;
int promien = 16;

while(1)
{
	if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
		stan_aktualny = 1;
	else
		stan_aktualny = 0;

	if(stan_aktualny != stan_poprzedni)
	{
		if(stan_aktualny == 1)
		{
			BSP_LCD_Clear(LCD_COLOR_WHITE);
			wynik_rzutu = rand() %6 + 1;
			if(wynik_rzutu == 1)
			{
				BSP_LCD_FillCircle(120, 160, promien);
			}
			if(wynik_rzutu == 2)
			{
				BSP_LCD_FillCircle(80, 120, promien);
				BSP_LCD_FillCircle(160, 200, promien);
			}
			if(wynik_rzutu == 3)
			{
				BSP_LCD_FillCircle(80, 120, promien);
				BSP_LCD_FillCircle(120, 160, promien);
				BSP_LCD_FillCircle(160, 200, promien);
			}
			if(wynik_rzutu == 4)
			{
				BSP_LCD_FillCircle(80, 120, promien);
				BSP_LCD_FillCircle(80, 200, promien);
				BSP_LCD_FillCircle(160, 120, promien);
				BSP_LCD_FillCircle(160, 200, promien);
			}
			if(wynik_rzutu == 5)
			{
				BSP_LCD_FillCircle(80, 120, promien);
				BSP_LCD_FillCircle(80, 200, promien);
				BSP_LCD_FillCircle(120, 160, promien);
				BSP_LCD_FillCircle(160, 120, promien);
				BSP_LCD_FillCircle(160, 200, promien);
			}
```

```
                    if(wynik_rzutu == 6)
                    {
                        BSP_LCD_FillCircle(80, 120, promien);
                        BSP_LCD_FillCircle(80, 160, promien);
                        BSP_LCD_FillCircle(80, 200, promien);
                        BSP_LCD_FillCircle(160, 120, promien);
                        BSP_LCD_FillCircle(160, 160, promien);
                        BSP_LCD_FillCircle(160, 200, promien);
                    }
                }
            }
            stan_poprzedni = stan_aktualny;
            HAL_Delay(50);
}
```

7. Test refleksu