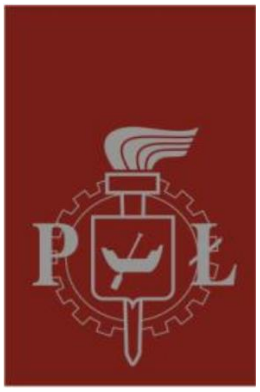


Politechnika Łódzka
Instytut Automatyki

Laboratorium Robotów Usługowych

Instrukcja 1

Processing

Łódź 2021

Wstęp

Celem tej części laboratorium jest zapoznanie się z podstawami języka Processing oraz funkcjonalnościami biblioteki controlP5 umożliwiającej obsługę kontrolek GUI. Instrukcja zawiera sposób konfiguracji środowiska oraz podstawową składnię języka.

Processing (licencja GPL) jest to język programowania oparty na Javie znajdujący zastosowanie w sztuce elektronicznej (wizualnej, interaktywnej, dźwiękowej) i projektowaniu grafiki. Ze względu na prostą składnię język wykorzystywany jest do nauki programowania oraz przez profesjonalistów, którzy chcą zwiększyć swoją produktywność.

Najczęściej używane kontrolki zostały opisane w instrukcji i dostępne są wraz z przykładami. Więcej kontrolek można znaleźć w dokumentacji: http://www.sojamo.de/libraries/controlP5/

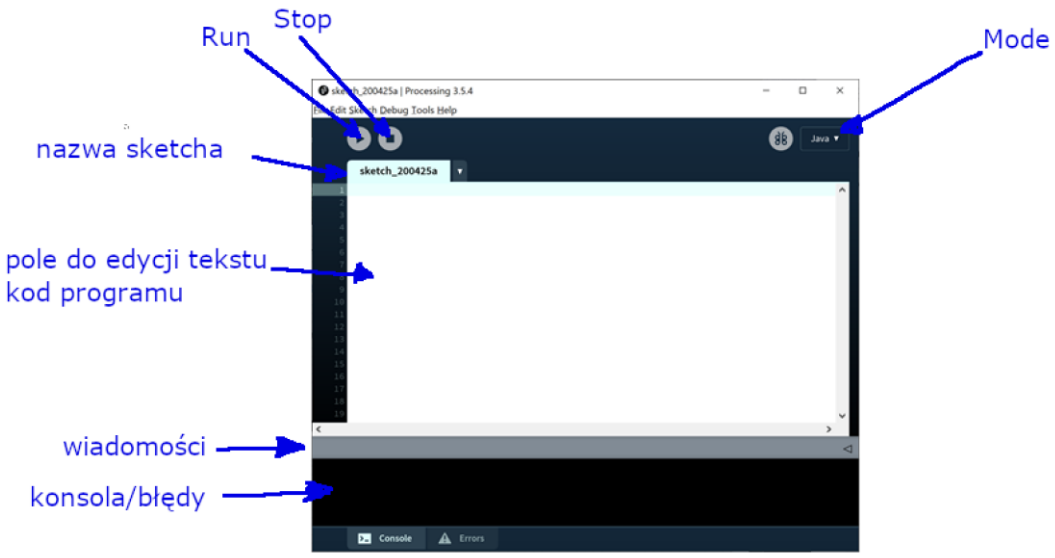
Dodatkowe przykłady i polecenia:

Link do przykładów https://processing.org/examples/

Link do wszystkich poleceń https://processing.org/reference/

Pierwsze uruchomienie i konfiguracja Processing

Uruchamiamy aplikację Processing IDE. Domyślnie w folderze *Documents* utworzony został domyślny folder, w którym przechowywane są pliki sketch.



Struktura plików Sketch

W ramach laboratorium każdy z tworzonych plików musi zawierać następujące dwie metody:

- setup() – Instrukcja wykonywana tylko raz przy starcie podczas inicjalizacji zmiennych i ustawień. Mogą to być zmienne dotyczące rozmiaru okna, czcionki, kolorów i innych parametrów.
- draw() – Instrukcja odpowiada głównej pętli programu. Odpowiada za wyświetlanie informacji na ekranie, interakcję z użytkownikiem i reaguje na zdarzenia w czasie rzeczywistym. Domyślnie odświeżanie ekranu wynosi 60 fps.

Powyższe metody są bezparametrowe i nie zwracają wartości.

Pierwszy program

Uruchomienie pierwszego programu pozwoli na weryfikację poprawności konfiguracji środowiska.

```
void setup() {
  fullScreen();
  noStroke();
  fill(0);
}

void draw() {
  background(204);
  if (mousePressed) {
    if (mouseX < width/2) {
      rect(0, 0, width/2, height); // Left
    } else {
      rect(width/2, 0, width/2, height); // Right
    }
  }
}
```

Pry kopiowaniu programu może pojawić się błąd ze znakami. Najprawdopodobniej będzie dotyczył występujących spacji przed poleceniami w każdej linii. Po uruchomieniu programu (Run) ekran powinien zrobić się szary. Program należy uruchomić w trybie Java jak i Android, aby sprawdzić czy wszystko działa tak jak należy Po uruchomieniu aplikacji w trybie Android następuje jej instalacja na telefonie i pojawia się ikona z nazwą programu taką samą jak nazwa pliku sketch. W zależności od tego po której stronie kliknięcie jest aktywne wskazana połowa ekranu podświetla się na czarno (lewa lub prawa połowa).

W Processing następujące zmienne domyślnie przechowują następujące wartości i nie trzeba się zastanawiać w jaki sposób odebrać te dane od urządzenia:

- mouseX – położenie kursora X
- mouseY – położenie kursora Y
- width – szerokość ekranu
- height – wysokość ekranu
- mousePressed – sprawdza czy przycisk myszy jest wciśnięty

Metoda *rect()* przyjmuje kolejno następujące argumenty współrzędna x, współrzędna y, szerokość i wysokość prostokąta.

Dokumentacja języka dostępna jest na stronie: https://processing.org/reference/

Więcej przykładów można znaleźć na stronie: https://processing.org/examples/

Typy danych

Symbol	Opis	Przykład
boolean	typ logiczny, przyjmuje wartości true/false	zmienna = true;
byte	liczby 1-bajtowe	a = -1;
int	liczby całkowite	a = 1;
float	liczby 4-bajtowe zmiennoprzecinkowe	a = 1.0;
char	typ znakowy zapisywany w pojedynczym cudzysłowie	znak = ‘a’;

String	typ tekstowy (z wielkiej litery), zawiera łańcuch tekstowy. Przy deklaracji używamy podwójnego cudzysłowu.	tekst = „przykład”;

color	zmienna przechowująca kolor w postaci RGB (wartości od 0 do 255 każda) lub w postaci heksadecymalnej.	kolor = color(0,255,0);
		lub
		kolor = #00FF00;

Składnia

Tablice

Struktura

typ[] zmienna_tablicowa = new typ[liczba_elementow_tablicy];

Przykład

```
int wylosowane_liczby = new int[6]
```

Utworzona została zmienna tablicowa typu int o nazwie wylosowane_liczby 6 elementów.

Możliwe jest też definiownaie tablic wielowymiarowych

```
int wiersze = 5;
int kolumny = 3;
int[][] tablica2D = new int[wiersze][kolumny]
```

Operatory matematyczne i logiczne

Podobnie jak w innych językach (np. C++) do dyspozycji mamy podstawowe operatory matematyczne i logiczne. Dostępne są również operacje skrócone.

Operator	Operator skrócony	Opis
+	+=	dodawanie
-	-=	odejmowanie
/	/=	dzielenie
*	*=	mnożenie
%		dzielenie modulo
++		inkrementacja (zwiększenie o 1)
--		dekrementacja (zmniejszenie o 1)

Operatory logiczne

Operator	Opis
<	mniejsze od
<=	mniejsze równe
>	większe
>=	większe równe
==	równe
!=	nierówne, różne
&&	logiczne AND
||	logiczne OR
!	logiczne NOT

Instrukcje warunkowe

Instrukcja if. Należy pamiętać o nawiasie, w którym znajduje się warunek.

```
if (warunek1)
```

```
{
    …       //instrukcje do wykonania, gdy warunek1 jest prawdziwy
}
else
{
   …        //instrukcje do wykonania, gdy warunek1 jest fałszywy
};
```

Instrukcja switch. Należy pamiętać o nawiasie, w którym znajduje się warunek.

```
switch (zmienna)
{
    case wartosc_1:
        … //ciag instrukcji gdy zmienna ma wartosc_1;
        break;
    case wartość_2:
        … //ciag instrukcji gdy zmienna ma wartość_2;
    default:
        … //ciag instrukcji dla pozostałych wartości zmiennej;
}
```

Pętle

```
for(inicjalizacja_zmiennej; warunek; krok)
{
  … //ciag instrukcji
}
```

Przykład

```
for(int i = 5; i < 10; i = i + 2)
{
  … //ciag instrukcji
}
```

Funkcje

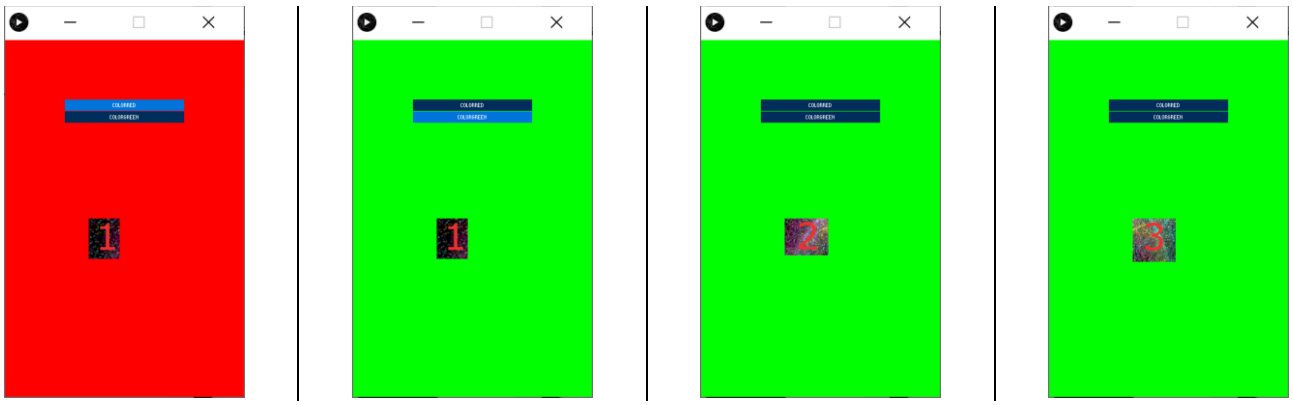
Użytkownik ma możliwość zdefiniowania własnych funkcji przyjmujących określoną liczbę argumentów i zwracających odpowiedni typ danych. Jeśli nic nie jest zwracane to funkcja jest typu `void`.

```
typ nazwa_funkcji(typ1 parametr_1;typ2 parametr_2; ...;typn parametr_n)
{
    … //ciąg instrukcji;
    return otrzymany_wynik_dzialania_funkcji;
}
```

GUI

Gotowe przykłady do uruchomienia znajdują się w załączniku do instrukcji

Przyciski



Dwa górne przyciski odpowiadają za zmianę tła ekranu na czerwony lub zielony kolor. Natomiast wygląd dolnego przycisku zmodyfikowany jest przy użyciu grafiki png. W zależności od akcji kursora zmienia wygląd ikony. Kursor znajduje się poza ikoną to wyświetlana jest pierwsza grafika, kursor najechał na ikonę to druga, a gdy nastąpiło wciśnięcie przycisku to wyświetlana jest ikona 3-cia.

Jednym ze sposobów na wywołanie konkretnej akcji dla danego przycisku jest utworzenie funkcji o tej samej nazwie co kontrolka i obsłużenie jej tak jak to ma miejsce w zaprezentowanym przykładzie. Parametry konfiguracyjne kontrolki można podawać jako wartości kolejnych argumentów zgodnie z dostarczoną dokumentacją lub ustawiać wartości konkretnych pól z wykorzystaniem metod. Polecana jest druga opcja, gdyż pozwala lepiej i szybciej przeanalizować działanie programu.

Dla przycisku zmieniającego grafikę wszystkie pliki png z nim związane znajdują się na tym samym poziomie lokalizacji plików co plik uruchomieniowy od processingu.

Plik z przykładem w dodatku do instrukcji: „buttons”

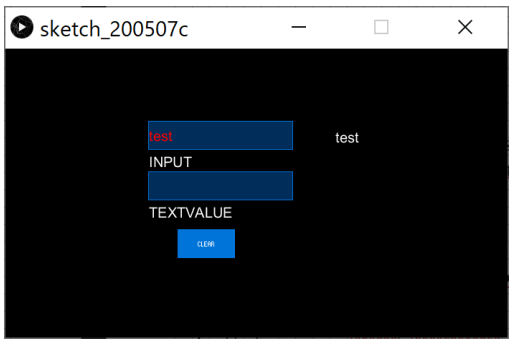
Kod programu

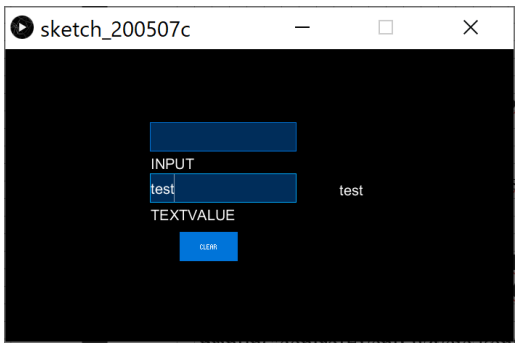
```
import controlP5.*; //importowanie biblioteki od kontrolek

ControlP5 cp5;
color currentColor = color(0,0,0);

void setup() {
  size(400,600);
  noStroke();
  cp5 = new ControlP5(this);

  cp5.addButton("colorRed") //utworzenie kontrolki button o nazwie colorRed
     .setValue(0) //ustawienie wartosci kontrolki na 0
     .setPosition(100,100) //polozenie kontrolki na ekranie
     .setSize(200,19) //rozmiar kontrolki
     ;

  cp5.addButton("colorGreen")
     .setPosition(100,120)
     .setSize(200,19)
     ;

  PImage[] imgs = {loadImage("button1.PNG"),loadImage("button2.PNG"),loadImage("button3.PNG")};
//zmienna tablicowa przechowujaca nazwy grafik
  cp5.addButton("play") //utworzenie przycisku play
     .setValue(128)
     .setPosition(140,300)
```

```
      .setImages(imgs)  //ustawienie zdjec jako wyglad przycisku
      .updateSize()     //zmiana rozmiwaru przycisku, dopasowanie do zdjecia
      ;
}

void draw() {
  background(currentColor);
}

public void controlEvent(ControlEvent theEvent) { //wywołanie funkcji, gdy zarejestrowano akcje
zwiazana z dowolna kontrolka
  println(theEvent.getController().getName());
}

// function colorRed will receive changes from
// controller with name colorRed
public void colorRed(int theValue) {
  println("a button event from colorRed: "+theValue);
  currentColor = color(255,0,0);
}

// function colorGreen will receive changes from
// controller with name colorGreen
public void colorGreen(int theValue) {
  println("a button event from colorGreen: "+theValue);  //wyswietlenie wartosci przekazanej
przez przycisk
  currentColor = color(0,255,0);
}
```

Obsługa pól tekstowych

1. INPUT - Pole do wprowadzania tekstu. Zatwierdzenie następuje po wciśnięciu przycisku enter. W trakcie wprowadzanina znaków obok wyświetlany jest tekst. Po zatwierdzeniu tekst jest przekazywany do wyświetlenia w konsoli. Znika wprowadzony tekst wpisany na interfejsie
2. TEXTVALUE – tekst wprowadzany jest podobnie jak dla pola INPUT. Tym razem po zatwierdzeniu wartość tekstowa nie znika z pola. Wyświetlana jest obok na interfejsie wprowadzona wartość po zatwierdzeniu enterem.
3. CLEAR – przycisk do czyszczenia tekstu wpisanego w polu TEXTVALUE. Nie czyści natomiast zmiennej, w której przechowywany jest ostatnio wprowadzony tekst z tego pola.





Plik z przykładem w dodatku do instrukcji: „text_fields”

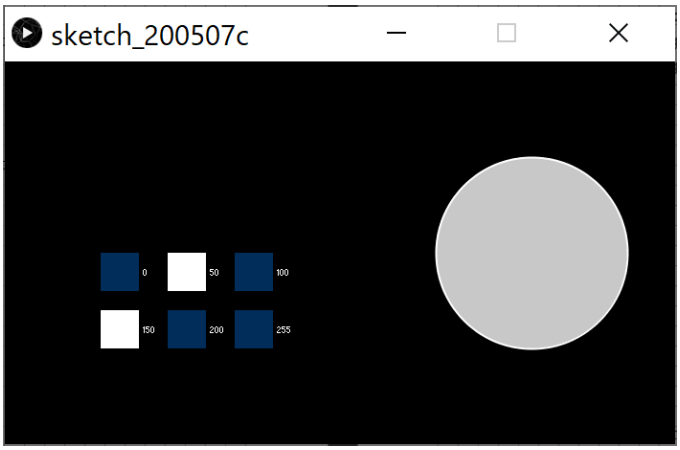
Kod programu

```
import controlP5.*;

ControlP5 cp5;
String textValue = "";

void setup() {
  size(700,400);
```

```
  PFont font = createFont("arial",20);

  cp5 = new ControlP5(this);

  cp5.addTextfield("input") //utworzenie pola tekstowego o nazwie input
     .setPosition(200,100)  //ustawienie polozenia pola tekstowego x,y
     .setSize(200,40)       //ustawienie rozmiaru pola tekstowego
     .setFont(font)         //ustawienie czcionki
     .setColor(color(255,0,0))  //ustawienie koloru wpisywanego tekstu
     ;

  cp5.addTextfield("textValue")
     .setPosition(200,170)
     .setSize(200,40)
     .setFont(createFont("arial",20))  //zamiast wykorzystania zmiennej od razu wywołana metoda
do ustawienia typu i rozmiaru czcionki
     .setAutoClear(false)       //wyłączenie natychmiastowego czyszczenia pola po zatwierdzeniu
enterem wprowadzonego tekstu
     ;

  cp5.addBang("clear")      //utworzenie przycisku o nazwie "clear" do czyszczenia tekstu w polu
TEXTVALUE
     .setPosition(240,250)
     .setSize(80,40)
     .getCaptionLabel().align(ControlP5.CENTER, ControlP5.CENTER)  //umieszczenie na srodku
przycisku jego nazwy
     ;
  textFont(font);
}

void draw() {
  background(0);
  fill(255);
  text(cp5.get(Textfield.class,"input").getText(), 460,130); //odbieranie wartosci z pola typu
Textfield o nazwie "input" wprowadzonego tekstu i wyswietlanie go
  text(textValue, 460,200); //wyswietlanie ostatnio wprowadzonego i zatwierdzonego tekstu w polu
TEXTVALUE
}

public void clear() {// funkcja wywolana po wcisnieciu przycisku o nazwie clear, ma taka sama
nazwe jak nazwa przycisku
  cp5.get(Textfield.class,"textValue").clear(); //czyszczenie wprowadzonego tekstu w polu, ale
nie nastepuje zmiana ostatnio zapamietanej wartosci dotyczacej tekstu
}

void controlEvent(ControlEvent theEvent) { //odebrano akcje pochodzaca od dowolnego przycisku
  if(theEvent.isAssignableFrom(Textfield.class)) { //jesli przypisano zdarzenie pochodzace od
przycisku typu Textfield
    println("controlEvent: uzycie kontrolera '"
            +theEvent.getName()+"' wartosc: "
            +theEvent.getStringValue()
            ); //wyswietlenie informacji w konsoli o zrodle zdarzenia i wartosci wpisanej w polu
  }
}


public void input(String theText) {//funkcja wywolywana po wprowadzeniu i zatwierdzeniu tekstu
enterem
  // automatically receives results from controller input
  println("wartosc pola 'input' : "+theText);
}
```

Obsługa checkboxów

Kolejnym ważnym elementem jest możliwość wyboru przez użytkownika kilku z dostępnych opcji np. do filtrowania wyświetlania ważnych dla użytkownika informacji.



Plik z przykładem w dodatku do instrukcji: „checkbox”

Kod programu

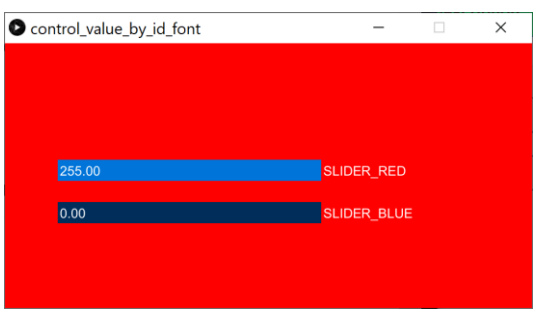
```
/**
 * ControlP5 Checkbox
 * an example demonstrating the use of a checkbox in controlP5.
 * CheckBox extends the RadioButton class.
 * to control a checkbox use:
 * activate(), deactivate(), activateAll(), deactivateAll(), toggle(), getState()
 *
 * find a list of public methods available for the Checkbox Controller
 * at the bottom of this sketch's source code
 *
 * by Andreas Schlegel, 2012
 * www.sojamo.de/libraries/controlP5
 *
 */
import controlP5.*;
ControlP5 cp5;
CheckBox checkbox;
int myColorBackground;

void setup() {
  size(700, 400);
  smooth();
  cp5 = new ControlP5(this);
  checkbox = cp5.addCheckBox("checkBox")  //checkbox, pole wielokrotnego wyboru
                .setPosition(100, 200) //poczatkowe polozenie pola
                .setColorForeground(color(120)) //ustaienie koloru dla niewybranej opcji
                .setColorActive(color(255)) //ustawienie koloru dla zaznaczonej opcji
                .setColorLabel(color(255))
                .setSize(40, 40)  //rozmiar checkboxa
                .setItemsPerRow(3) //liczba opcji w wierszu
                .setSpacingColumn(30) //odstep pomiedzy kolumnami
                .setSpacingRow(20) //odstep pomiedzy kolejnymi wierszami opcji
                .addItem("0", 0) //dodawanie nowych opcji nazwa,wartosc
                .addItem("50", 50)
                .addItem("100", 100)
                .addItem("150", 150)
                .addItem("200", 200)
                .addItem("255", 255)
                ;
}
```

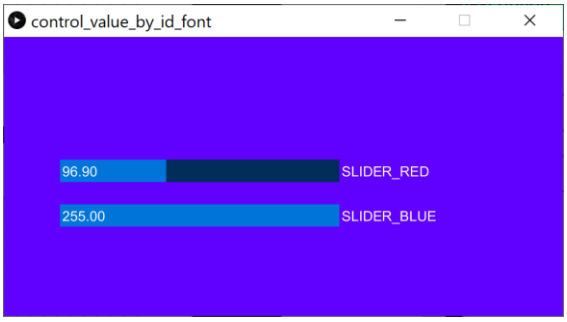
```
void keyPressed() { //obsluga akcji z przyciskow klawiatury
  if (key==' ') { //wcisnieta spacja
    checkbox.deactivateAll(); //odznaczenie wszystkich zaznaczonych pol
  }
  else {
    for (int i=0;i<6;i++) {
      // check if key 0-5 have been pressed and toggle
      // the checkbox item accordingly.
      if (keyCode==(48 + i)) {
        // the index of checkbox items start at 0
        checkbox.toggle(i);
        println("toggle "+checkbox.getItem(i).getName());
        // also see
        // checkbox.activate(index);
        // checkbox.deactivate(index);
      }
    }
  }
}

void draw() {
  background(0);
  pushMatrix();
  translate(width/2 + 200, height/2);
  stroke(255);
  strokeWeight(2);
  fill(myColorBackground);
  ellipse(0,0,200,200);
  popMatrix();
}

void controlEvent(ControlEvent theEvent) { //obsluga zdarzen od przyciskow
  if (theEvent.isFrom(checkbox)) { //sprawdzenie warunku czy zdarzenie pochodzi od obiektu
checkbox
    myColorBackground = 0;
    print("got an event from "+checkbox.getName()+"\t\n");
    // checkbox uses arrayValue to store the state of
    // individual checkbox-items. usage:
    println(checkbox.getArrayValue()); //odbior tablicy wartosci kolejnych pol
    int col = 0;
    for (int i=0;i<checkbox.getArrayValue().length;i++) { //petla odpowiadajaca za wyswietlanie
kolejnych wartsci tablicy wraz z informacja czy pole jest zaznaczone (1) czy tez nie (0(
      int n = (int)checkbox.getArrayValue()[i];
      print(n);
      if(n==1) {
        myColorBackground += checkbox.getItem(i).internalValue(); //odebranie wartosci pola dla
checkboxa o numerze i
      }
    }
    println();
  }
}

void checkBox(float[] a) {
  println(a);
}
```

Odbiór wartości z kontrolek po ich ID, ustawienie rozmiaru czcionki, slider

Innym sposobem na obsługę kontrolek jest przypisanie im ściśle określonego ID. W zależności od tego, która jest aktualnie używana mogą być podjęte inne akcje. Przykładowy program ustawia intensywność koloru czerwonego i niebieskiego w tle programu w zależności od położenia suwaków. Domyślny rozmiar czcionki i cyfr na ekranie może

nie być czytelny, dlatego zostały one powiększone. Na poprzednich przykładach można było zauważyć, że czcionka była zbyt mała i trudno było odczytać jakiekolwiek wartości.





Plik z przykładem w dodatku do instrukcji: „control_value_by_id_font”

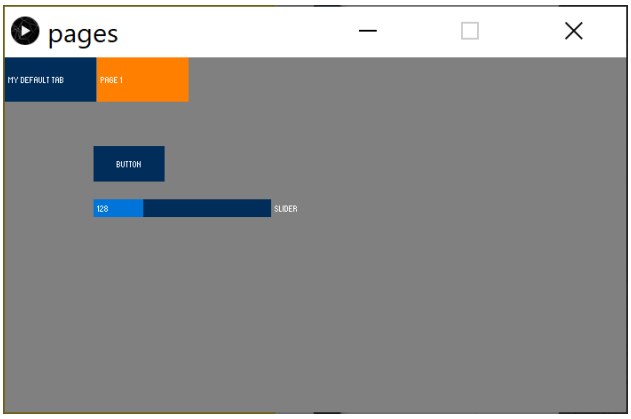
Kod programu

```
import controlP5.*;

ControlP5 cp5;

int redColor = 255;
int blueColor = 255;

void setup() {
  size(1000,500);
  noStroke();
  PFont pfont = createFont("Arial",20,true); // use true/false for smooth/no-smooth
  ControlFont font = new ControlFont(pfont,25); //obiekt klasy PFont, drugi argument to rozmiar

  /* new instance of ControlP5 */
  cp5 = new ControlP5(this);
  /* add 2 controllers and give each of them a unique id. */
  cp5.addSlider("slider_red")
     .setRange(0,255)
     .setValue(redColor)
     .setPosition(100,220)
     .setSize(500,40)
     .setFont(font) //ustawienie czcionki dla tej kontrolki
     .setId(1); //id kontrolki

  cp5.addSlider("slider_blue")
     .setRange(0,255)
     .setValue(blueColor)
```

```
      .setPosition(100,300)
      .setSize(500,40)
      .setFont(font)
      .setId(2);
}

void draw() {
  background(redColor,0,blueColor);
}

void controlEvent(ControlEvent theEvent) {
  /* events triggered by controllers are automatically forwarded to
     the controlEvent method. by checking the id of a controller one can distinguish
     which of the controllers has been changed.
  */
  println("Slider id event "+theEvent.getController().getId()); //wyswietlenie id aktualnie uzywanego kontrolera w konsoli Processingu
  switch(theEvent.getController().getId()) {
    case(1):
    /* controller slider1 with id 1 */
    redColor = (int)theEvent.getValue(); //przypisanie wartosci ze slidera 1 do zmiennej redColor; rzutowanie wartości do int bo takie wartości przyjmują kolory, a ze slidera mogą wychdzić wartości zmiennoprzecinkowe
    break;
    case(2):
    /* controller slider1 with id 2 */
    blueColor = (int)theEvent.getValue();
    break;
  }
}
```

Obsługa zakładek

Kolejnym elementem umożliwiającym zarządzaniem i uporządkowaniem wielu przycisków jest możliwość pogrupowania ich i wyświetlania tylko potrzebnych kontrolek w zależności od kontekstu. Na przykład jedna z nich może odpowiadać formularzowi do wprowadzania listy zadań do wykonania, a druga z nich będzie umożliwiała ich wyświetlenie.



Plik z przykładem w dodatku do instrukcji: „pages”

Kod programu

```
import controlP5.*;

ControlP5 cp5;
int myColorBackground = color(128);
int sliderValue = 100;

void setup() {
  size(700,400);
  noStroke();
  cp5 = new ControlP5(this);

  // By default all controllers are stored inside Tab 'default'
  // add a second tab with name 'extra'


  cp5.addTab("page_2")
     .setColorBackground(color(0, 160, 100))
     .setColorLabel(color(255))
     .setColorActive(color(255,128,0))
     .activateEvent(true)
     .setLabel("page 2")
     .setId(2)
     .setHeight(50)
     .setWidth(100)
     ;

  // if you want to receive a controlEvent when
  // a  tab is clicked, use activeEvent(true)

  cp5.getTab("default")
     .activateEvent(true)
     .setLabel("my default tab")
     .setHeight(50)
     .setWidth(100)
     .setId(1)
     ;
  cp5.getTab("page_2");
  // create a few controllers
  cp5.addButton("button")
     .setBroadcast(false)
     .setPosition(100,100)
     .setSize(80,40)
     .setValue(1)
     .setBroadcast(true)
     .getCaptionLabel().align(CENTER,CENTER)
     ;
```

```
  cp5.addButton("buttonValue")
     .setBroadcast(false)
     .setPosition(220,100)
     .setSize(80,40)
     .setValue(2)
     .setBroadcast(true)
     .getCaptionLabel().align(CENTER,CENTER)
     ;

  cp5.addSlider("slider")
     .setBroadcast(false)
     .setRange(100,200)
     .setValue(128)
     .setPosition(100,160)
     .setSize(200,20)
     .setBroadcast(true)
     ;


  // arrange controller in separate tabs

  cp5.getController("button").moveTo("page_2"); //przypisanie elementu button do karty page_2
  cp5.getController("slider").moveTo("global"); //przypisanie elementu slider do wszystkich kart
  // przycisk buttonValue domyslnie przypisany jest do zakladki default i nie ma koniecznosci
ponownego
  //jego przypisania do karty
  // Tab 'global' is a tab that lies on top of any
  // other tab and is always visible

}

void draw() {
  background(myColorBackground);
  //fill(sliderValue);
}

void controlEvent(ControlEvent theControlEvent) {
  if (theControlEvent.isTab()) {
    println("got an event from tab : "+theControlEvent.getTab().getName()+" with id
"+theControlEvent.getTab().getId());
  }
}

void slider(int theColor) { //zmienna theColor przechowuje informacje o wartosci odczytanej ze
slidera
  myColorBackground = color(theColor);
  println("a slider event. setting background to "+theColor);
}


void keyPressed() {
  if(keyCode==TAB) {
    cp5.getTab("page_2").bringToFront(); //programowe przelaczenie do innej zakladki. Jesli jest
aktywna karta default
    //to po wcisnieciu tab nastepuje przelaczenie na karte page_2
  }
}
```

Zadania do wykonania

System wielorobotowy będzie się składał z 3 robotów (tb3_0, tb3_1, tb3_2).

Zadanie na 3

Wykorzystując język Processing zaprojektować interfejs użytkownika do obsługi 1 robota. Powinien on umożliwiać wybranie punktu (co najmniej 1 z 3), do którego ma dojechać robot oraz przycisk potwierdzający zlecenie zadania. Informacja o wysłaniu zadania powinna pojawić się na interfejsie.

Zadanie na 4

Rozbudować interfejs z zadania 3 o kontrolkę umożliwiającą wybór 1 z 3 robotów. do danego zadania. Utworzyć interfejs z poziomu, którego będzie się wybierało robota, którym użytkownik chce sterować. Jeśli robot jest w trakcie wykonywania zadania dojazdu do punktu to taka akcja będzie niemożliwa. Sterowanie może odbywać się z wykorzystaniem sliderów. Kontrolki powinny umożliwiać zmianę prędkości postępowej (max +- 0.26m/s) i obrotowej (max +- 1.82 rad/s)

Zadanie na 5

Połączyć utworzone wcześniej interfejsy w jeden z wykorzystaniem zakładek. Dołożyć kontrolki, które informowałyby o tym czy robot wykonuje w danej chwili zadanie czy nie. Można wyświetlić tekst informujący o zachowaniu wszystkich robotów, jeśli jadą do jakiegoś celu to wyświetlany jest cel, a jeśli nie mają go to pojawia się informacja, że robot jest wolny.