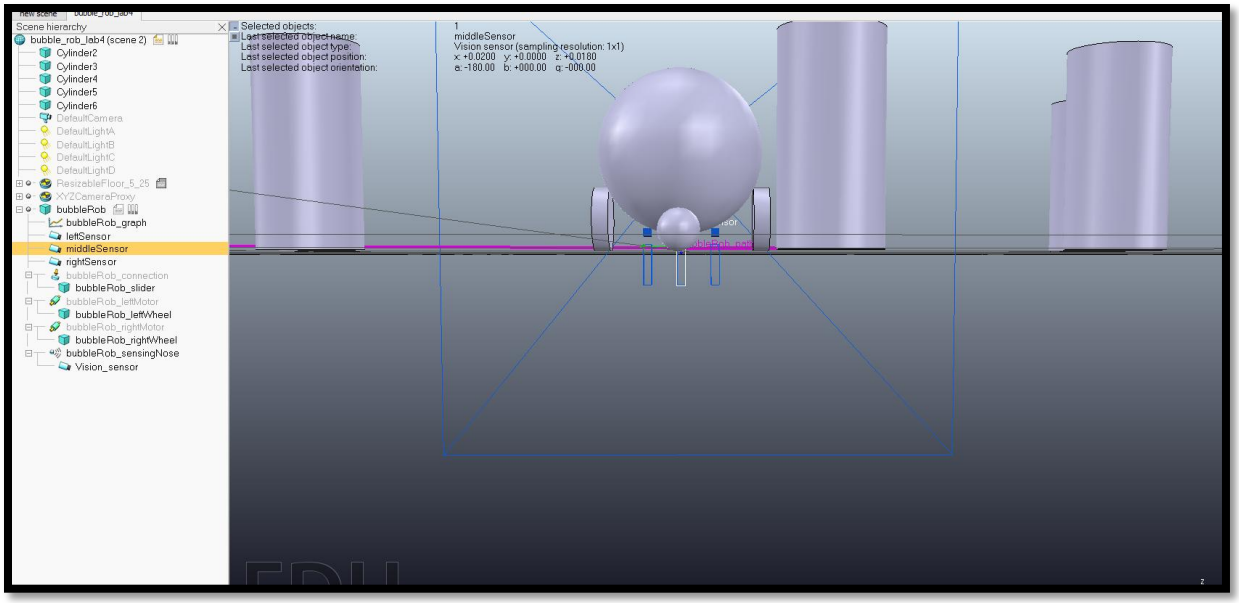
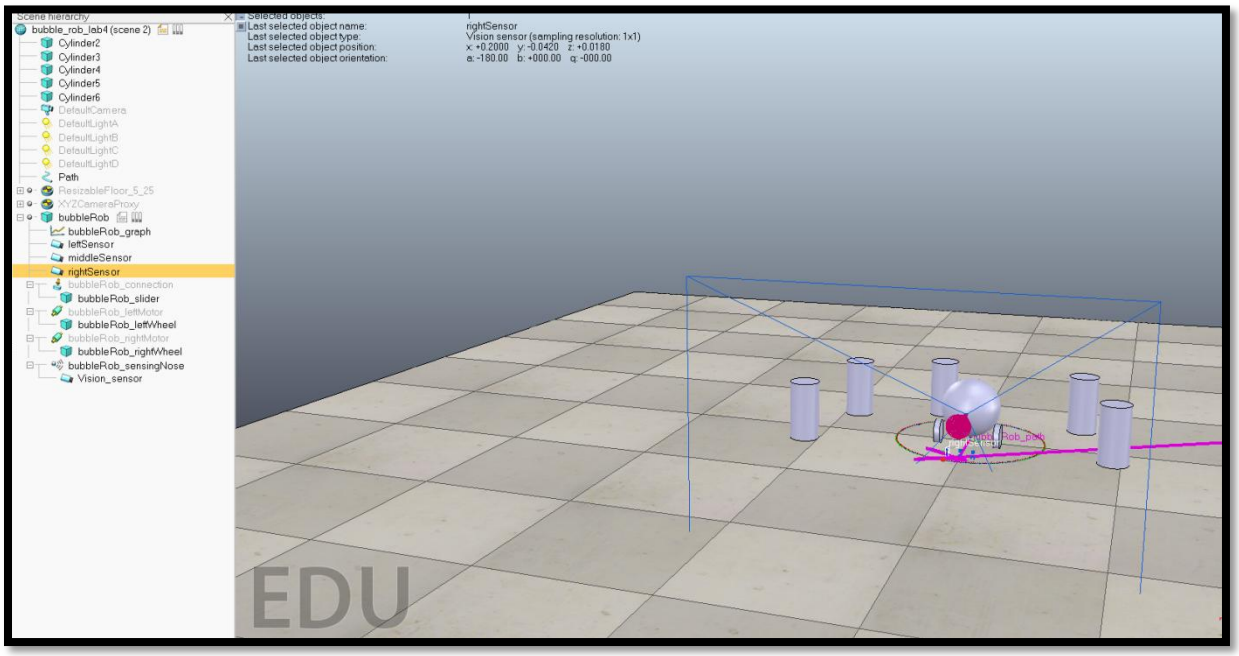
Sprawozdanie Lab 4

Andrzej Żaba, gr_lab 4, nr indeksu: 401490

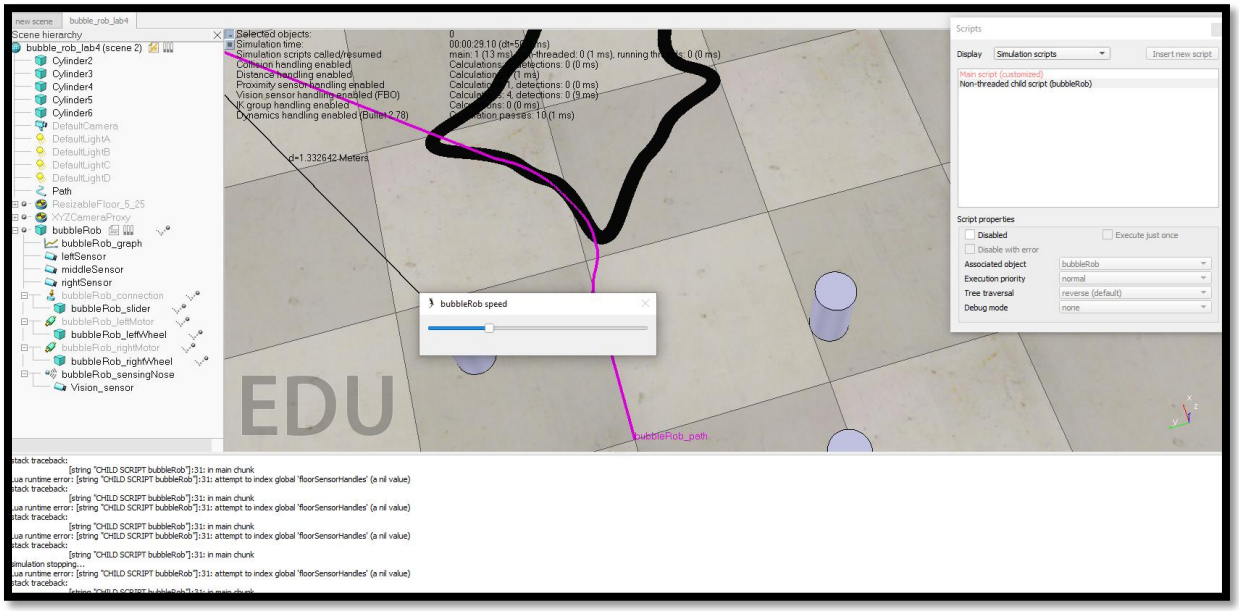
Zaczęto od otworzenia sceny z bubbleRob-em, stworzonej na laboratorium 1. Dodano trzy czujniki wizyjne oraz nadano im odpowiednią pozycję i orientację. Podpięto je w drzewku modelu pod bazę modelu, czyli element **bubbleRob**, skutkuje to utrzymaniem względnej pozycji czujników w czasie ruchu robota.



Następnie usunięto kilka cylindrów oraz narysowano pierwszą ścieżkę.



Ścieżce ustawiono kolor na czarny, skalowanie na 4,00 oraz zmieniono:
„**dwuklik na ikonę ścieżki w drzewie hierarchii -> Show path shaping dialog -> Section characteristics: Type -> horizontal segment**”



Następnie zajęto się modyfikacją skryptu robota:
„**ikona na drzewku modelu przy elemencie: bubbleRob**”.

Pisanie kodu zaczęto od zainicjowania zmiennych i podpięcia czujników pod jedną z nich:

```
floorSensorHandles = {-1,-1,-1}
sensorReading={false, false, false}

floorSensorHandles[1] = sim.getObjectHandle("leftSensor")
floorSensorHandles[2] = sim.getObjectHandle("middleSensor")
floorSensorHandles[3] = sim.getObjectHandle("rightSensor")
```

tablica *floorSensorHanldes[]* przechowuje w sobie dodane wcześniej czujniki wizyjne.
sensorReading natomiast będzie przechowywać odczyty z owych czujników. Jest to tablica zmiennych typu **Boolean**. Celem działania robota jest poruszanie się po stworzonej linii. Zadaniem czujników natomiast jest wykrycie, czy znajdują się nad czarną linią, czy też nie – zmienna typu *prawda / fałsz* będzie to tego odpowiednia.

Następnie zajęto się kodem w instrukcji warunkowej:

```
if (sim_call_type==sim.syscb_actuation) then
  sensorReading={false, false, false}
```

Owa instrukcja warunkowa jest czymś w rodzaju odpowiednika znanej np. z mikrokontrolerów pętli *while(1)* która odpowiada, za cykliczne wykonywanie głównych zadań programu.

Przy każdym obiegu pętli ustawiamy zmienne w tablicy *sensorReading* na *false*, ponieważ za każdym razem program będzie na bieżąco decydował jakie działania podjąć, dlatego koniecznym jest wyzerowanie wyników z poprzedniego obiegu.

Kolejno zadeklarowano pętlę for odczytującą stan czujników:

```
for i=1,3,1 do
    result,data = sim.readVisionSensor(floorSensorHandles[i])
    if(result>=0) then
      if data[11] <= 0.33 then          -- data[11] - average intensity
        sensorReading[i] = true
      end
    end
  end
```

Dla każdego z czujników następuje odczyt.
Jeżeli odczyt się powiódł, sprawdzane jest, czy intensywność odczytanego koloru mieści się w zadanym zakresie (roboczo przyjęto x <= 0.33).
Jeśli odczytana intensywność mieści się w zakresie, oznacza to, że czujnik znajduje się nad naszą linią, przypisujemy zmiennej wartość *true*.

Dalej napisano instrukcje warunkowe determinujące działanie silników robota w zależności od odczytów czujników wizyjnych.

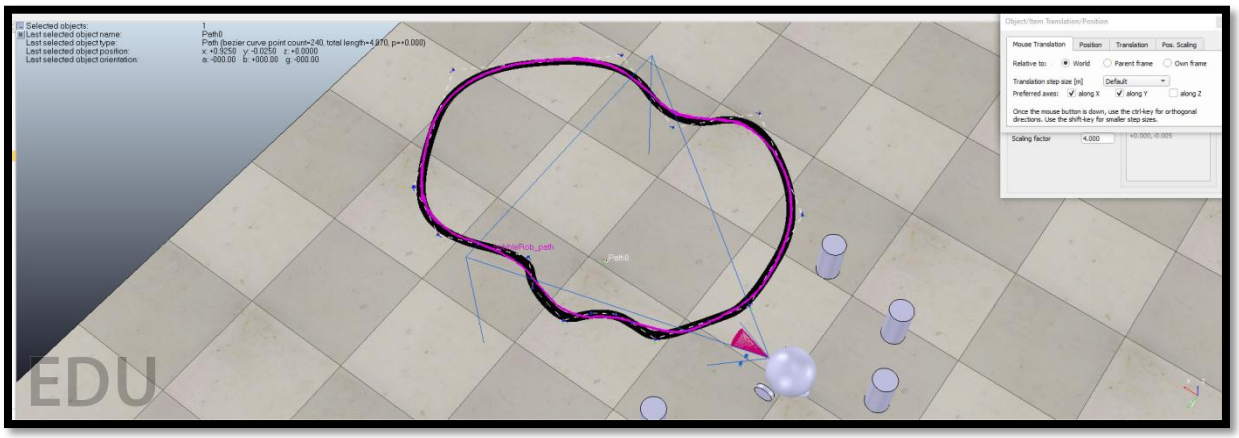
```
-- take action based on sensorReading
rightV = speed
leftV = speed

if sensorReading[1] then
   leftV = 0.03*speed
   rightV = 1.7*speed
end
if sensorReading[3] then
   rightV = 0.03*speed
   leftV = 1.7*speed
end
```

W zależności od odczytów sensorów robot będzie skręcał lekko w lewo lub prawo, aby utrzymać się na linii. Może również utrzymać bazową prędkość i jechać do przodu. W razie pozytywnego odczytu z lewego jak i prawego czujnika wprowadzono zabezpieczenie, przed przejechaniem trasy na wskroś. Opiera się ono na wycofaniu i lekkim obrocie robota, aby próbował odnaleźć ścieżkę pod innym kątem, co ułatwi mu zadanie. Reszta kodu:

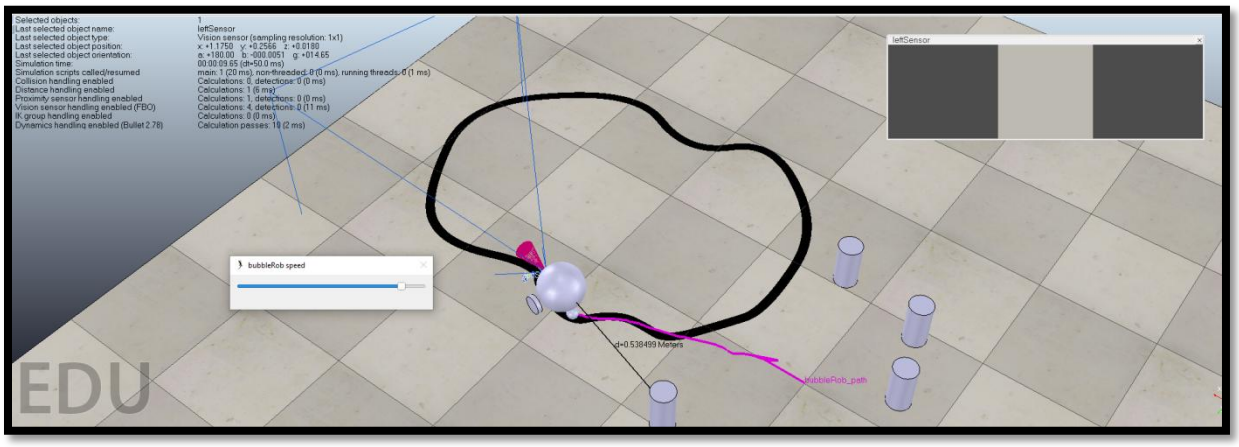
```
if sensorReading[1] and sensorReading[3] then
   backUntilTime=sim.getSimulationTime()+2
end
```

```
result=sim.readProximitySensor(noseSensor) -- Read the proximity sensor
-- If we detected something, we set the backward mode:
if (result>0) then backUntilTime=sim.getSimulationTime()+2 end
if (backUntilTime<sim.getSimulationTime()) then
-- When in forward mode, we simply move forward at the desired speed
sim.setJointTargetVelocity(leftMotor,leftV)
sim.setJointTargetVelocity(rightMotor,rightV)
else
-- When in backward mode, we simply backup in a curve at reduced speed
sim.setJointTargetVelocity(leftMotor,-speed/2)
sim.setJointTargetVelocity(rightMotor,-speed/8)
end
end
if (sim_call_type==sim.syscb_cleanup) then
simUI.destroy(ui)
end
```

Po wstępnych testach oraz poprawkach związanych z korektą ścieżki stwierdzono poprawność pracy robota.



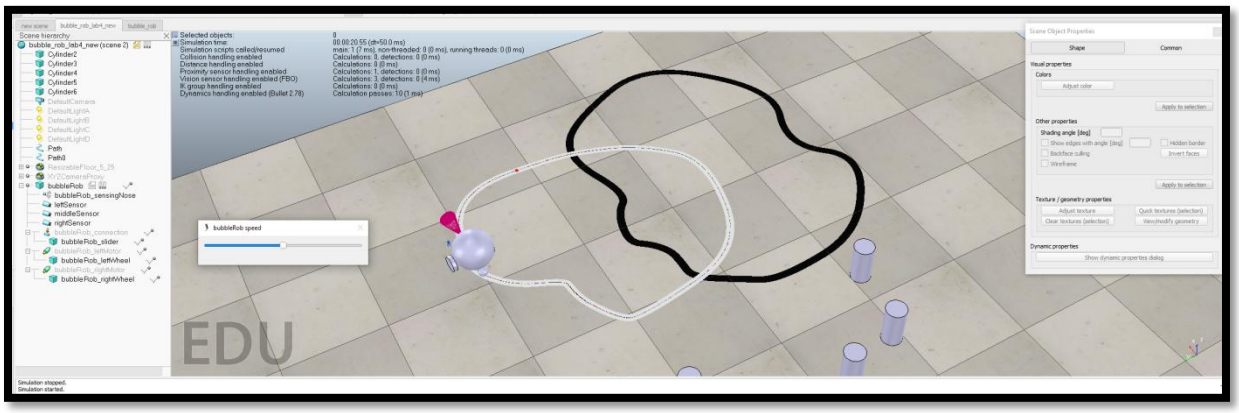
Następnie sprawdzono na bieżąco odczyty z jednego z czujników: „zaznaczyć sensor w drzewie modelu **->** w przestrzeni roboczej kliknąć prawym przyciskiem myszy **-> Add -> Floating view ->** prawy przycisk myszy na oknie dialogowym **-> View -> Assosiate view with selected Visio sensor**”



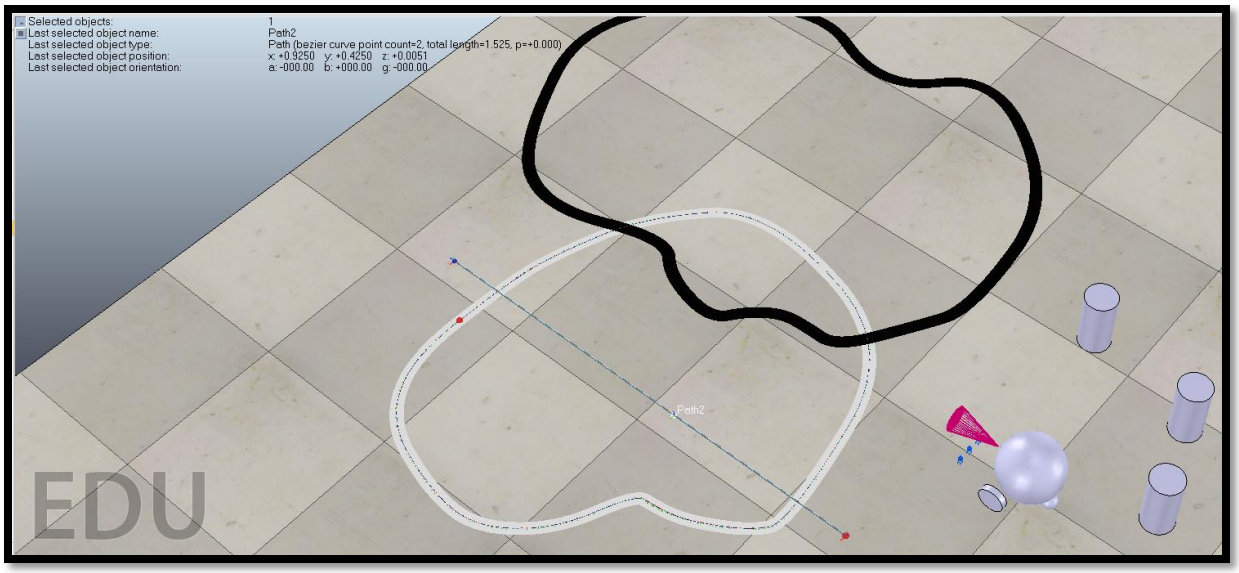
Następnie usunięto elementy takie jak kamera z przodu robota, rysowanie przebytej trasy, miara odległości od najbliższego obiektu. Dodano również białą linię, w podobny sposób jak uprzednio czarną. W części inicjacji wartości w skrypcie dodano zmienną:

```
colorMode = 2      -- 1 - black | 2 – white
```

Jest to zmienna, którą określić należy bezpośrednio w skrypcie. Odpowiada ona za tryb w którym ma działać nasz robot – śledzić czarną czy białą linię.



Dodano do sceny również szarą linię (kolory RGB po 50 i skalowanie 0.5, typ: vertical segment)



Jest to prowizoryczne rozwiązanie kwestii zatrzymania się robota w danym miejscu – robot zatrzyma się na 2 sekundy po najechaniu na szarą linię.

Aby wprowadzić wspomniane funkcjonalności do skryptu robota należało zmodyfikować pętlę odpowiadającą za odczyt wartości czujników w następujący sposób:

```
for i=1,3,1 do
    result,data = sim.readVisionSensor(floorSensorHandles[i])
    if(result>=0) then
        if colorMode == 1 then
        -- Dark Mode
            if data[11] <= 0.33 then-- data[11] - average intense / intensity
                sensorReading[i] = true
            end
        else
        -- White Mode
            if data[11] > 0.90 then -- data[11] - average intensity
                sensorReading[i] = true
            end
        -- Waitnig on Grey Line
            if i == 2 and data[11] > 0.4 and data[11] <0.6 then
                wait(2)
            end
        end
    end
end
```

Natomiast funkcję *wait()* zdefiniowano następująco:

```
function wait(seconds)
   local start = os.time()
   repeat until os.time() > start + seconds
end
```