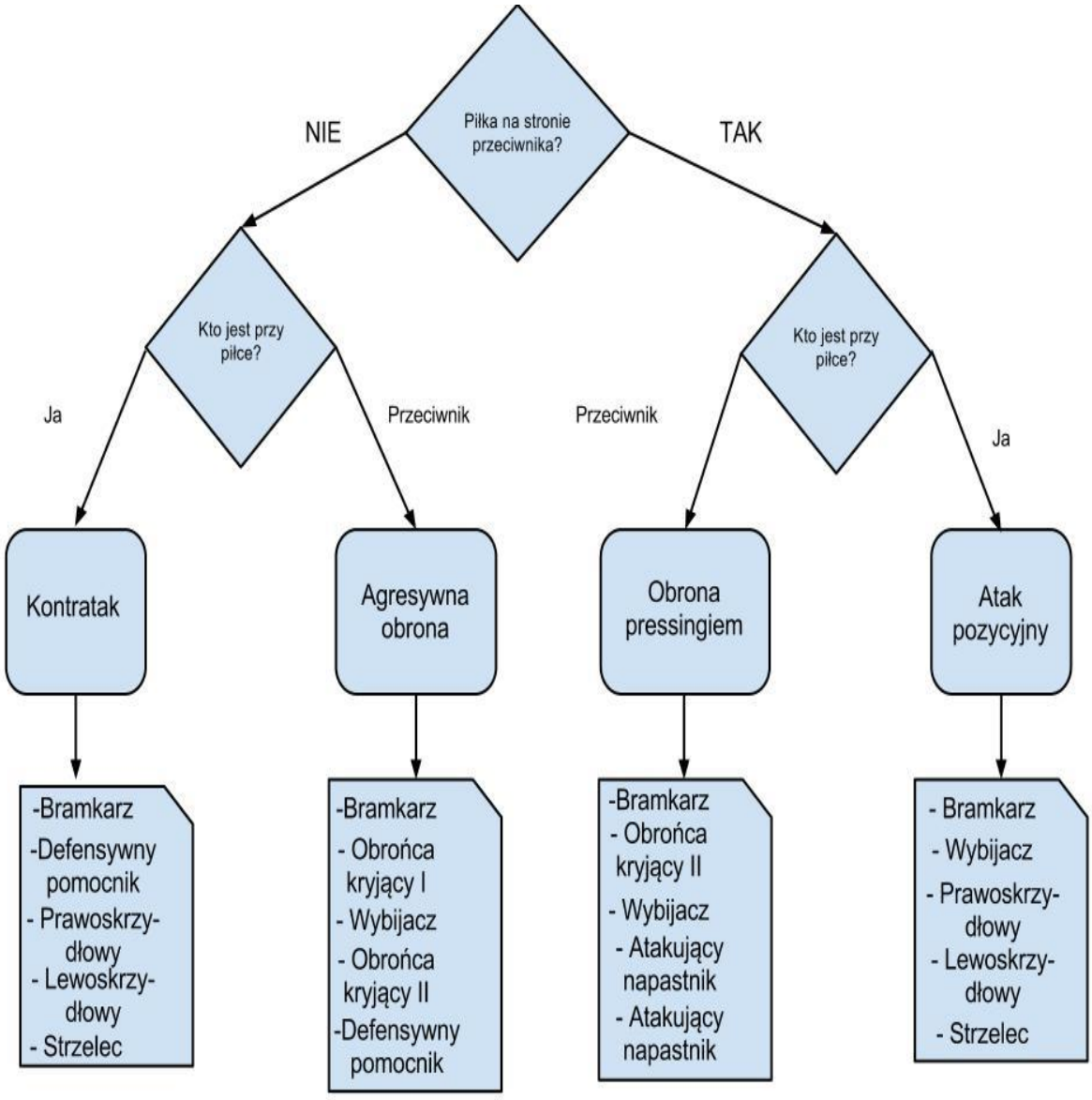
Tadeusz Wieja

Sprawozdanie numer 3

1. Graf strategii.



2. Opis poszczególnych składowych.

- **Warstwa I**

  W warstwie numer I metoda gdzie_pilka() definiuje na, której połowie boiska znajduje się piłka. Sprawdzenie to odbywa się poprzez zmienną currentBall.pos.x obiektu typu Environment. Następnym etapem jest wywołanie metody kto_przy_pilce(), która sprawdza, która z drużyn znajduje się przy piłce poprzez

ustalenie, który z robotów znajduje się najbliżej piłki. De dwa parametry determinują wybór odpowiedniej strategii chwilowej.

- **Warstwa II**

  W każdej strategii chwilowej Bramkarz nie jest brany pod uwagę w priorytetach wyboru. Bramkarzem jest zawsze robot o ID = 0.

  i. **Strategia chwilowa „Kontratak" -** celem jest szybkie przejście na stronę rywala i oddanie strzału. Role w strategii: bramkarz, defensywny pomocnik, prawoskrzydłowy, lewoskrzydłowy, strzelec. Algorytm podziału ról:
     a. Bramkarz – przypisany na stałe do robota o ID = 0.
     b. Defensywny pomocnik – Robot najbliżej bramki za wyjątkiem robota o ID=0 (brana pod uwagę tylko współrzędna X), nieposiadający piłki.
     c. Prawoskrzydłowy – Robot znajdujący się najbliżej prawej bandy boiska, nieposiadający piłki.
     d. Lewoskrzydłowy – Robot znajdujący się najbliżej lewej strony boiska, nie posiadający piłki.
     e. Strzelec – Robot znajdujący się najbliżej piłki.

     Priorytety wyboru:

     1) Strzelec.
     2) Lewoskrzydłowy.
     3) Prawoskrzydłowy.
     4) Defensywny pomocnik.

  ii. **Strategia chwilowa „Agresywna obrona" -** celem jest przeszkadzanie rywalowi w oddaniu strzału. Role w strategii: bramkarz, obrońca kryjący I, obrońca kryjący II, wybijacz, defensywny pomocnik. Algorytm podziału ról:
     a. Bramkarz – przypisany na stałe do robota o ID = 0.
     b. Wybijacz – Robot najbliżej środka linii pola karnego (brana pod uwagę współrzędna X i Y).
     c. Obrońca kryjący I – Robot znajdujący się najbliżej lewej bandy boiska.
     d. Obrońca kryjący II – Robot znajdujący się najbliżej prawej bandy boiska.
     e. Defensywny pomocnik – Robot znajdujący się najbliżej środkowej linii.

     Priorytety wyboru:

     1) Obrońca kryjący I.
     2) Obrońca kryjący II.
     3) Wybijacz.
     4) Defensywny pomocnik.

iii. **Strategia chwilowa „Obrona pressingiem" -** celem jest szybki odbiór piłki na połowie przeciwnika. Role w strategii: bramkarz, obrońca kryjący II, wybijacz, 2x atakujący napastnik. Algorytm podziału ról:
   a. Bramkarz – przypisany na stałe do robota o ID = 0.
   b. Wybijacz – Robot najbliżej środka linii pola karnego (brana pod uwagę współrzędna X i Y).
   c. Obrońca kryjący II – Robot znajdujący się najbliżej prawej bandy boiska.
   d. Atakujący napastnik – Roboty znajdujący się najbliżej bramki przeciwnika (brana pod uwagę tylko współrzędna X).

   Priorytety wyboru:

   1) Atakujący napastnik.
   2) Obrońca kryjący.
   3) Wybijacz.

iv. **Strategia chwilowa „Atak pozycyjny"** – celem jest oddanie strzału. Role w strategii: bramkarz, wybijacz, prawoskrzydłowy, lewoskrzydłowy, strzelec. Algorytm podziału ról:
   a. Bramkarz – przypisany na stałe do robota o ID = 0.
   b. Wybijacz – Robot najbliżej bramki za wyjątkiem robota o ID=0 (brana pod uwagę tylko współrzędna X), nieposiadający piłki.
   c. Prawoskrzydłowy – Robot znajdujący się najbliżej prawej bandy boiska, nieposiadający piłki.
   d. Lewoskrzydłowy – Robot znajdujący się najbliżej lewej strony boiska, nie posiadający piłki.
   e. Strzelec – Robot znajdujący się najbliżej piłki.

   Priorytety wyboru:

   1) Strzelec.
   2) Lewoskrzydłowy.
   3) Prawoskrzydłowy.
   4) Wybijacz.

- **Warstwa III**
   i. **Bramkarz -** zawodnik o ID = 0, poruszający się w linii bramkowej w zależności od strony boiska (lewa, prawa) gdzie znajduje się piłka.
   ii. **Defensywny pomocnik –** robot zmierza do linii środkowej (punkt x=43.3, y=35.5).
   iii. **Prawoskrzydłowy –** Zawodnik ofensywny znajdujący się po prawej stronie boiska. Zmierza on w kierunku prawego rogu pola karnego (punkt x=13.8, y=25.6) w celu czekania na ewentualną dobitkę.

iv. **Lewoskrzydłowy -** Zawodnik ofensywny znajdujący się po prawej stronie boiska. Zmierza on w kierunku lewego rogu pola karnego (punkt x=13.8, y=61.9) w celu czekania na ewentualną dobitkę.

v. **Strzelec –** Napastnik, robot posiadający piłkę. Porusza się w kierunku bramki, starając się być na wprost bramki. Po zbliżeniu się do pola karnego (punkt x=71, y= 35.5) oddaje strzał poprzez zatrzymanie się w miejscu.

vi. **Obrońca kryjący I –** Obrońca znajdujący się po lewej stronie boiska (powyżej punktu y=62). Jeżeli przeciwnik posiadający piłkę jest po jego stronie atakuje go próbując zablokować (jako argument przyjmowany jest obiekt zwracany przez funkcje kto_przy_pilce(), który zawiera zmienną posiadającą informacje o stronie gdzie znajduje się piłka) . W przeciwnym razie podąża za najbliżej znajdującym się zawodnikiem po jego stronie(Informację tą przekazuje mu obiekt zwracany przez funkcję kto_przy_pilce()).

vii. **Obrońca kryjący II -** Obrońca znajdujący się po prawej stronie boiska (poniżej punktu y=26). Jeżeli przeciwnik posiadający piłkę jest po jego stronie atakuje go próbując zablokować (jako argument przyjmowany jest obiekt zwracany przez funkcje kto_przy_pilce(), który zawiera zmienną posiadającą informacje o stronie gdzie znajduje się piłka) . W przeciwnym razie podąża za najbliżej znajdującym się zawodnikiem po jego stronie(Informację tą przekazuje mu obiekt zwracany przez funkcję kto_przy_pilce()).

viii. **Wybijacz –** Obrońca czekający na środku linii pola karnego (punkt x=43, y= 35.5). W przypadku zmierzania piłki w jego kierunku próbuje ją wybić na połowę przeciwnika poprzez poruszanie się w kierunku zbliżającej się piłki.

ix. **Atakujący napastnik -** zawodnik poruszający się w kierunku zawodnika przeciwnika znajdującego się na swojej połowie, który posiada piłkę (jako argument przyjmowany jest obiekt zwracany przez funkcje kto_przy_pilce(), który zawiera zmienną posiadającą informacje kto ma piłkę)**.**

3. **Implementacja.**

```
bool gdzie_pilka(Environment *env)
{
	if(env->currentBall.pos.x < 43.3)
	{
		//Pilka na stronie zoltych
		return 1;
	}
	else
		return 0;
}

void bramkarz(Robot* r1, Environment* env)
{
	if(env->predictedBall.pos.y > GTOPY)
	{
		Position(r1, 8, GTOPY);
	}
	else if(env->predictedBall.pos.y < GBOTY)
	{
		Position(r1, 8, GBOTY);
	}
	else
```

```
        {
                Position(r1, 8, env->predictedBall.pos.y);

        }
}

  class Roboty
  {
  public:
          unsigned int id;
          double odleglosc;
          bool pilka; // 1 przy pilce, 0 brak pilki
          bool kolor; // 1-zolty, 0-niebieski;
          bool strona; // 1 - lewa strona, 0 - prawa strona
          Roboty(void);
          ~Roboty(void);
          void zeruj();
  };
  Roboty::Roboty()
  {
          id = 0;
          pilka = 0;
          kolor = 0;
          strona = 0;
          odleglosc = 1000;
  }

  Roboty::~Roboty(void)
  {
  }

  void Roboty::zeruj()
  {
          pilka = 0;
          kolor = 0;
          strona = 0;
          odleglosc = 1000;
  }
  class boisko
  {
          bool strona; // 1 - lewa strona, 0 - prawa strona
          bool pilka; // 1 - zolty, 0 - niebieski
          double niebieski_close; // odleglosc najbliższego
          niebieskiego pilki
          double zolty_close; // odleglosc najbliższego zoltego pilki
          Roboty* niebiescy; // Informacje o druzynie niebieski jak
          daleko poszczegolnie zawodnicy sa od pilki
          Roboty* zolci; // Informacje o druzynie zoltych jak daleko
          poszczegolnie zawodnicy sa od pilki
          unsigned int niebieski_id; // id najblizszego niebieskiego
          unsigned int zolty_id; // id najblizszego zoltego
  public:
          boisko(void);
          ~boisko(void);
          void kto_przy_pilce(Environment *env);
  };
  boisko::boisko(void)
  {
          strona = 0;
          pilka = 0;
          niebiescy = new Roboty [4];
```

```
        zolci = new Roboty [4];
        niebieski_close = 1000;
        niebieski_id = 0;
        zolty_close = 1000;
        zolty_id = 0;
        for(int i=0; i<4; i++)
        {
                niebiescy[i].id = i+1;
                zolci[i].id = i+1;
                niebiescy[i].kolor = 0;
                zolci[i].kolor = 1;
        }
}

boisko::~boisko(void)
{
        delete [] niebiescy;
        delete [] zolci;
}

void boisko::kto_przy_pilce(Environment *env)
{
        niebieski_close = 1000; // przygotowanie zmiennych do
        porownania
        zolty_close = 1000;

        double x_Ball = env->currentBall.pos.x;
        double y_Ball = env->currentBall.pos.y;
        if(env->currentBall.pos.y < 35.5) // pilka po prawej
        stronie boiska;
        {
                strona = 0;
        }
        else
        {
                strona = 1;
        }
        for(int i=0; i<4; i++) // Obliczanie odległości do pilki i
        strony poszczegolnych zawodnikow
        {
                niebiescy[i].zeruj();

                niebiescy[i].odleglosc = sqrt(pow((env-
                >opponent[i+1].pos.x - env-
                >currentBall.pos.x),2)+pow((env->opponent[i+1].pos.y
                - env->currentBall.pos.y),2));

                if(env->opponent[i+1].pos.y > 35.5) // Po lewej
                stronie
                        niebiescy[i].strona = 1;
                else
                        niebiescy[i].strona = 0;
                zolci[i].zeruj();

                zolci[i].odleglosc = sqrt(pow((env->home[i+1].pos.x
                - env->currentBall.pos.x),2)+pow((env-
                >home[i+1].pos.y - env->currentBall.pos.y),2));

                if(env->home[i+1].pos.y > 35.5) // Po lewej stronie
                        niebiescy[i].strona = 1;
                else
```

```
                niebiescy[i].strona = 0;
        }
        for(int i=0; i<4; i++) // kto najbliżej pilki z niebieskich
        i zoltych;
        {
                if(niebiescy[i].odleglosc < niebieski_close)
                {
                        niebieski_close = niebiescy[i].odleglosc;
                        niebieski_id = i;
                }
                if(zolci[i].odleglosc < zolty_close)
                {
                        zolty_close = zolci[i].odleglosc;
                        zolty_id = i;
                }
        }
        if(zolty_close< niebieski_close) // Pilka zoltych
        {
                pilka = 1;
                zolci[zolty_id].pilka = 1;
        }
        else  //Pilka niebieskich
        {
                pilka = 0;
                niebiescy[niebieski_id].pilka = 1;
        }
}
```