Laboration 3: Grafisk display

Laboration 3 består av tre uppgifter varav två behandlas utförligt utförligt i läroboken där de beskrivs i form av deluppgifter som succesivt måste klaras av innan själva laborationsuppgifterna kan lösas.

För att kunna genomföra laborationen måste du ha arbetat igenom avsnitt 5.4 i läroboken.

Laborationsuppgifterna baseras på följande exempel och uppgifter:

- Exempel 5.3 t.o.m 5.6, uppgifter 5.10 t.o.m 5.17.

Laborationsuppgift 3.1: Enkla geometrier (Uppgift 5.14 i läroboken)

- Uppgiften är att konstruera en funktion som ritar en polygon till grafikdisplayen.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

Laborationsuppgift 3.2: Spindeljakt (Uppgift 5.17 i läroboken)

Uppgiften är att skapa en liten spelapplikation (1 användare) där en ”spindel” kan manövreras med hjälp av en keypad för att fånga in en liten boll som rör sig autonomt över skärmen. Som förberedelse för denna uppgift ska du även ha gjort uppgift 5.15 i läroboken.

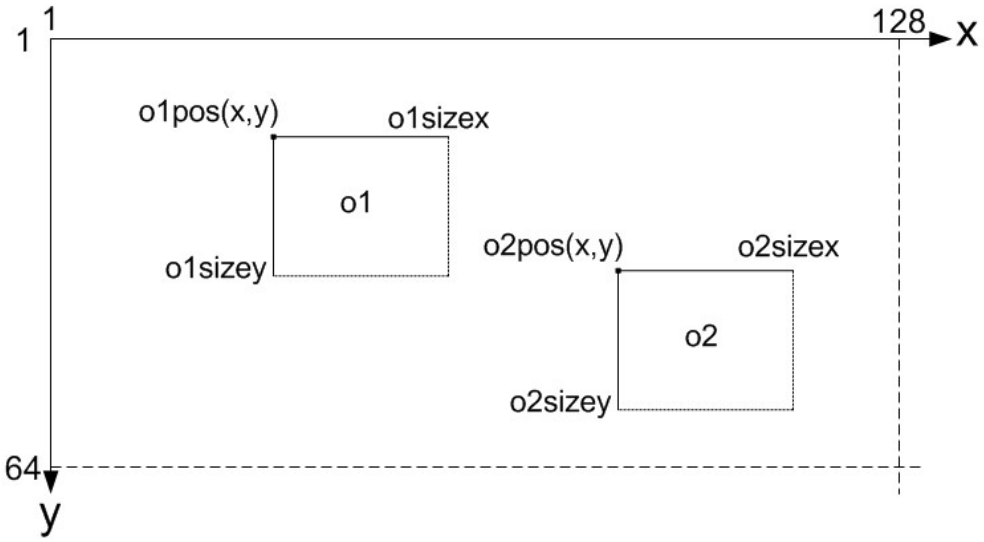
Metoder för att kontrollera överlapp mellan två objekt

Det intiutivt första försöket kan vara att jämföra de pixlar de respektive objekten ockuperar:

```
int  pixel_overlap(POBJECT o1, POBJECT o2)
{
  int offset1x = o1->posx;
  int offset1y = o1->posy;
  int offset2x = o2->posx;
  int offset2y = o2->posy;
  for( int i = 0; i < o1->geo->numpoints; i++ )
  {
    for( int j = 0; j < o2->geo->numpoints; j++ )
      if( ( offset1x+o1->geo->px[i].x == offset2x+o2->geo->px[j].x ) &&
          ( offset1y+o1->geo->px[i].y == offset2y+o2->geo->px[j].y ) )
             return 1;
  }
  return 0;
}
```

Metoden är exakt i den mening att bara ett geometriöverlapp kommer att detekteras. Som vi ser krävs `i * j` iterationer där `i` är antalet pixlar i det första objektet och `j` är antalet pixlar i det andra objektet. Det kan bli mycket exekveringstid, speciellt som kontrollen utförs minst en gång under varje tidscykel.

Som approximation kan man i stället jämföra objektens största utsträckningar i form av de största rektanglar som kan innesluta respektive objekts pixlar. Betrakta två objekt o1 och o2 med omslutande geometrier (ges i datastrukturen):



Av figuren framgår att vi med en approximerande funktion (med enkel komplexitet) snabbt kan undersöka om dessa båda rektanglar överlappar varandra genom att jämföra horisontella och vertikala gränsytor. Detta ger en betydligt snabbare funktion än den föregående och kan lämpligen användas här.

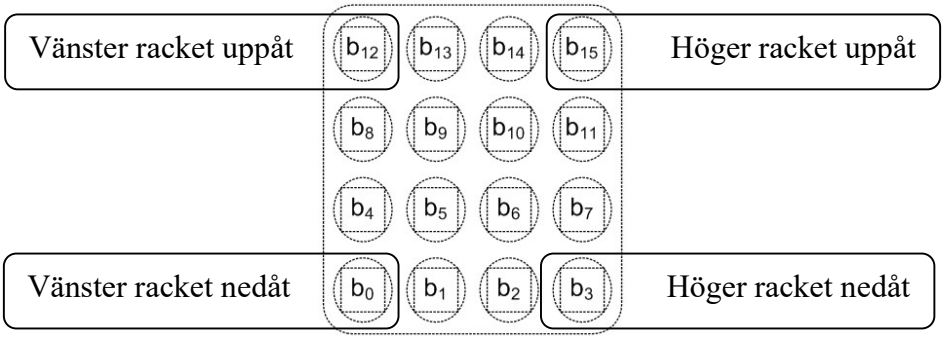
Man kan förfina genom att först använda en approximerande funktion och därefter (om så krävs) använda den exakta jämförelsen.

- Använd testprogrammet från lärobokens uppgift 5.17.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

Laborationsuppgift 3.3: Klassikern PONG för två spelare

I denna uppgift konstruerar du det klassiska tennisspelet PONG. Här ger vi ett minsta programskelett som löser uppgiften. Som förberedelse för denna uppgift ska du även ha gjort uppgift 5.16 i läroboken.

Det nya i denna uppgift är att vi nu använder två racketar i stället för en. Vi använder därför tangentbordsrutinen `keyb_alt_ctrl` från inlämning 1 där tangenterna nu ges följande betydelse:



Flera tangenter ignoreras alltså här men kan ges funktioner om man så önskar. Det finns möjligheter att utveckla detta senare under inlämning 5.

- Skapa ett nytt projekt `pong`.
- Modifiera rörelsefunktionerna för objekten, tänk på att bollen nu bara studsar mot de horisontella väggarna. Huvudprogrammet kontrollerar om racketen hindrar bollen från att röra sig ut över kanten.
- Använd huvudprogrammet till höger för att testa din PONG-applikation.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

```
int main()
{
  unsigned short s;
  POBJECT ball = &ball_object;
  POBJECT paddle1 = &paddle1_object;
  POBJECT paddle2 = &paddle2_object;
  init_app();
  graphic_initalize();
  graphic_clearScreen();
  ball->set_speed( ball, 4, 1);

  while( 1 )
  {
    ball->move( ball );
    paddle1->move( paddle1 );
    paddle2->move( paddle2 );
    s = keyb_alt_ctrl();

    if((s & (LEFT_UP | LEFT_DOWN))==(LEFT_UP|LEFT_DOWN))
      paddle1->set_speed( paddle1, 0, 0);
    else if (s & LEFT_UP )
      paddle1->set_speed( paddle1, 0, -4);
    else if (s & LEFT_DOWN )
      paddle1->set_speed( paddle1, 0, 4);
    else
      paddle1->set_speed( paddle1, 0, 0);

    if((s & (RIGHT_UP|RIGHT_DOWN))==(RIGHT_UP|RIGHT_DOWN))
      paddle2->set_speed( paddle2, 0, 0);
    else if (s & RIGHT_UP )
      paddle2->set_speed( paddle2, 0, -4);
    else if (s & RIGHT_DOWN )
      paddle2->set_speed( paddle2, 0, 4);
    else
      paddle2->set_speed( paddle2, 0, 0);

    if( objects_contact( ball, paddle1 ))
    {
      ball->clear(ball);
      ball->dirx = - ball->dirx;
      ball->draw(ball);
    }else if( ball->posx < (1 + ball->geo->sizex) )
    {
      break;
    }

    if( objects_contact( ball, paddle2 ))
    {
      ball->clear(ball);
      ball->dirx = - ball->dirx;
      ball->draw(ball);
    }else if( ball->posx > (128 - ball->geo->sizex) )
    {
      break;
    }
    delay_milli(40);
  }
}
```