TSEA83 : Datorkonstruktion
Fö7

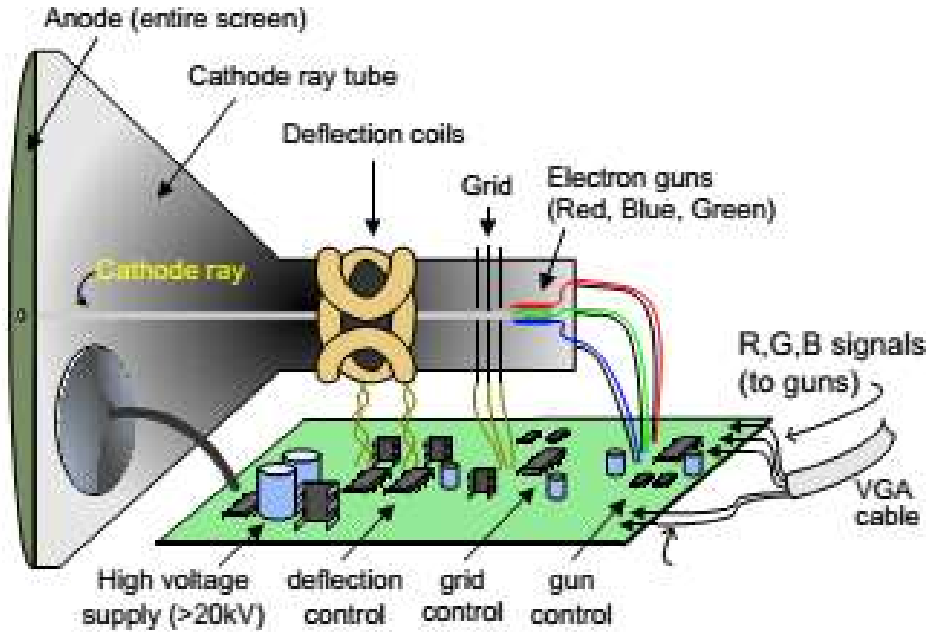
Grafik + Projekt

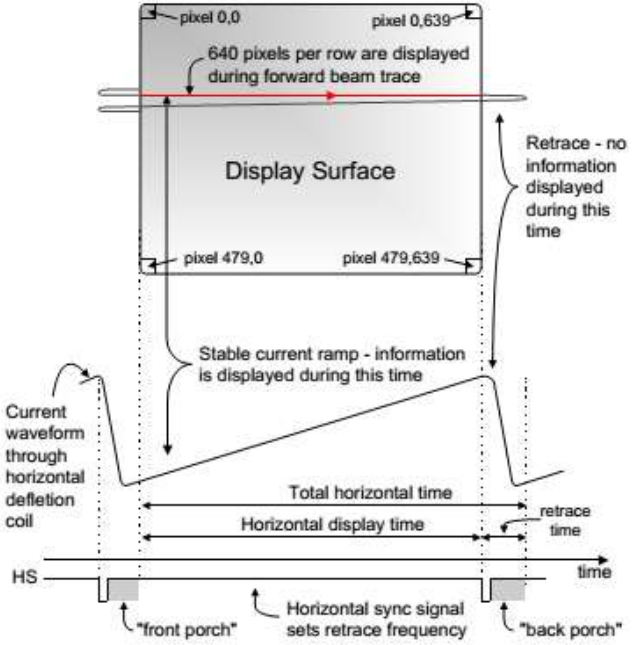
LINKÖPINGS UNIVERSITET

Fö7 : Agenda

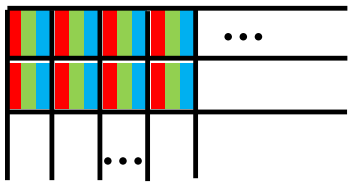
- Grafik förr
- VGA-signalen
- Direktdriven grafik eller bildminne
- Bitmap-grafik
- Tile/teckenbaserad grafik
- Spritebaserad grafik
- Kollisionskontroll
- Rörelse, hastighet/riktning
- Scrollande bakgrund/förgrund
- Labbar och projekt
- Projektdemo

Grafik ”förr”, typ tjock-TV





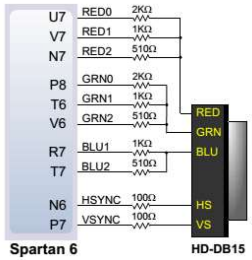
En elektronkanon skjuter elektroner genom spolarnas magnetfält. Elektronerna viker av olika mycket åt nåt håll beroende på magnetfältets styrka och träffar bildrutan i en punkt och lyser upp. Olika färg fås genom att träffa på olika delar (röd, grön eller blå del) inom punkten.

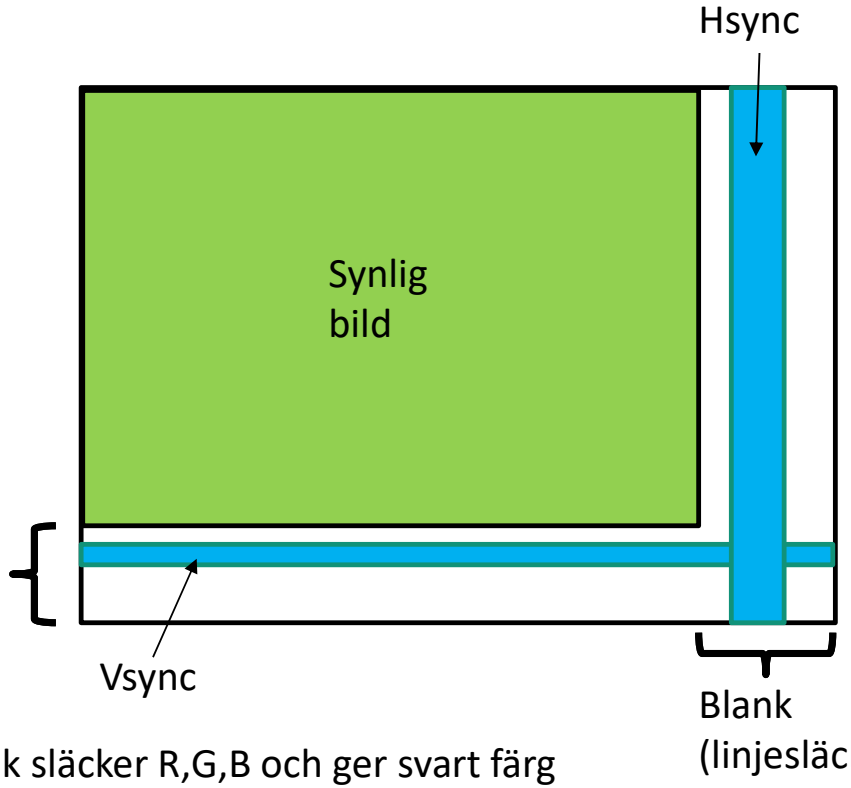


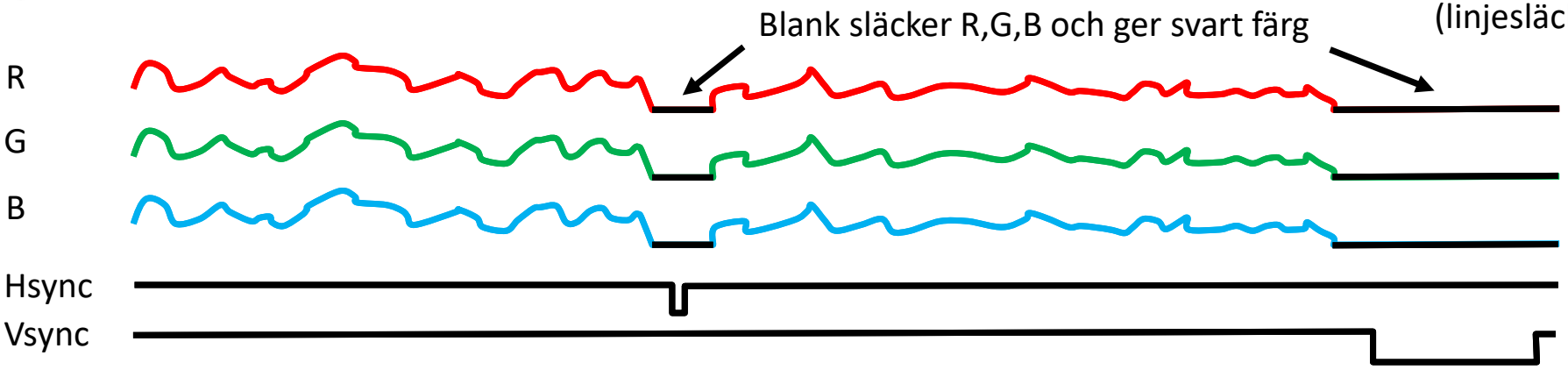
VGA-signalen

VGA-interfacet:

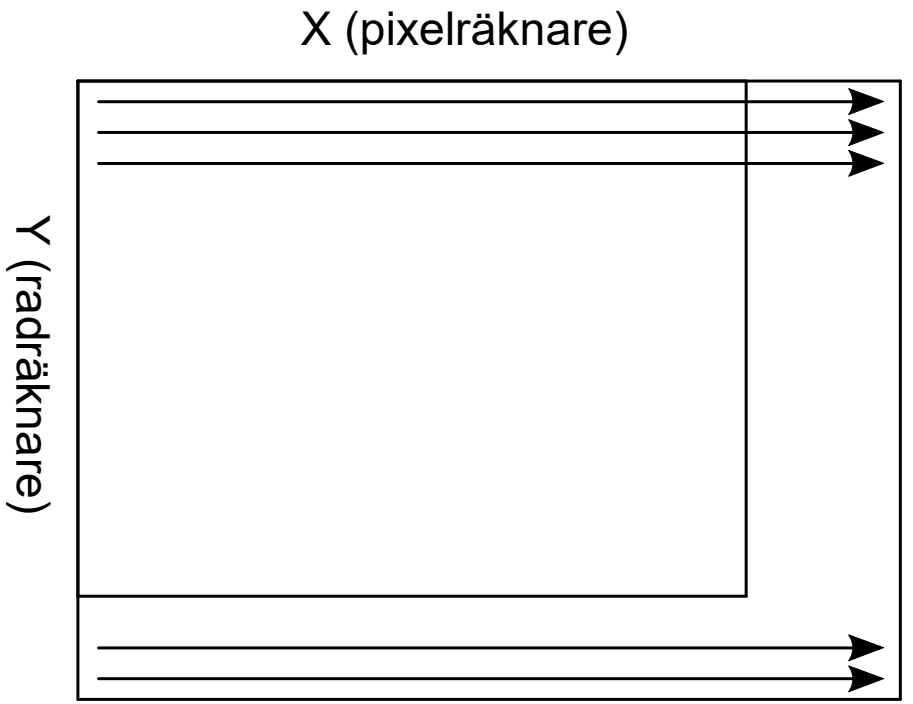
- Tre analoga signaler (Röd, Grön, Blå)
- Två synkpulser (vertikal och horisontal)
- (DDC2: EN I2C-buss för indentifiering av bildskärm)







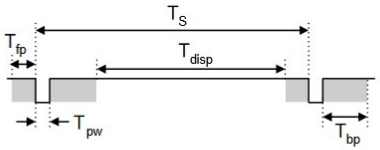
VGA-signalen



Alla signaler : Video (synlig bild, RGB), Hsync, Vsync, Blank kan skapas vid lämpliga värden på X och Y.

En viss upplösning uppnås via rätt timing, dvs rätt avstånd och längd på synkpulserna Hsync och Vsync.

Synktiming kan inte vara godtycklig, utan måste följa VGA-standard.



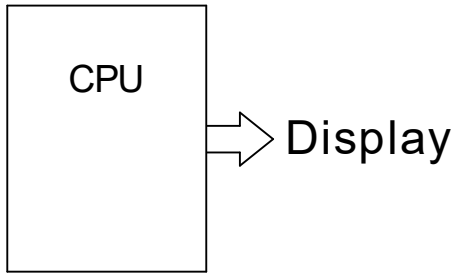
Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

Datablad för Basys3
→ 640 x 480 (synlig yta) i 60Hz bildfrekvens
→ Kräver 25 MHz pixelklocka

Direktdriven vs Bildminnesdriven grafik

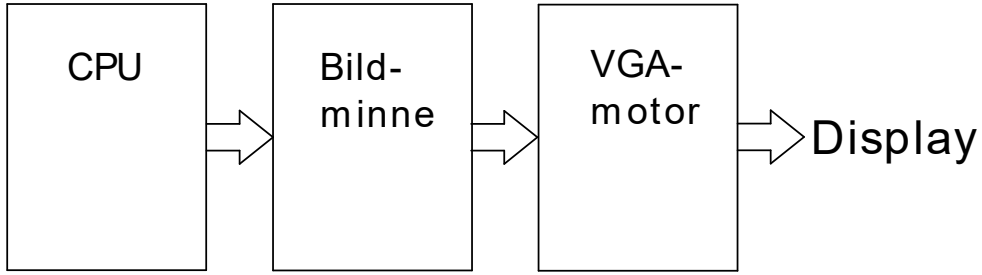
Direktdriven

- CPU:n måste leverera pixlar i exakt rätt takt
- Fördel : billigare hårdvara
- Nackdel : programmet blir extremt tidskritiskt

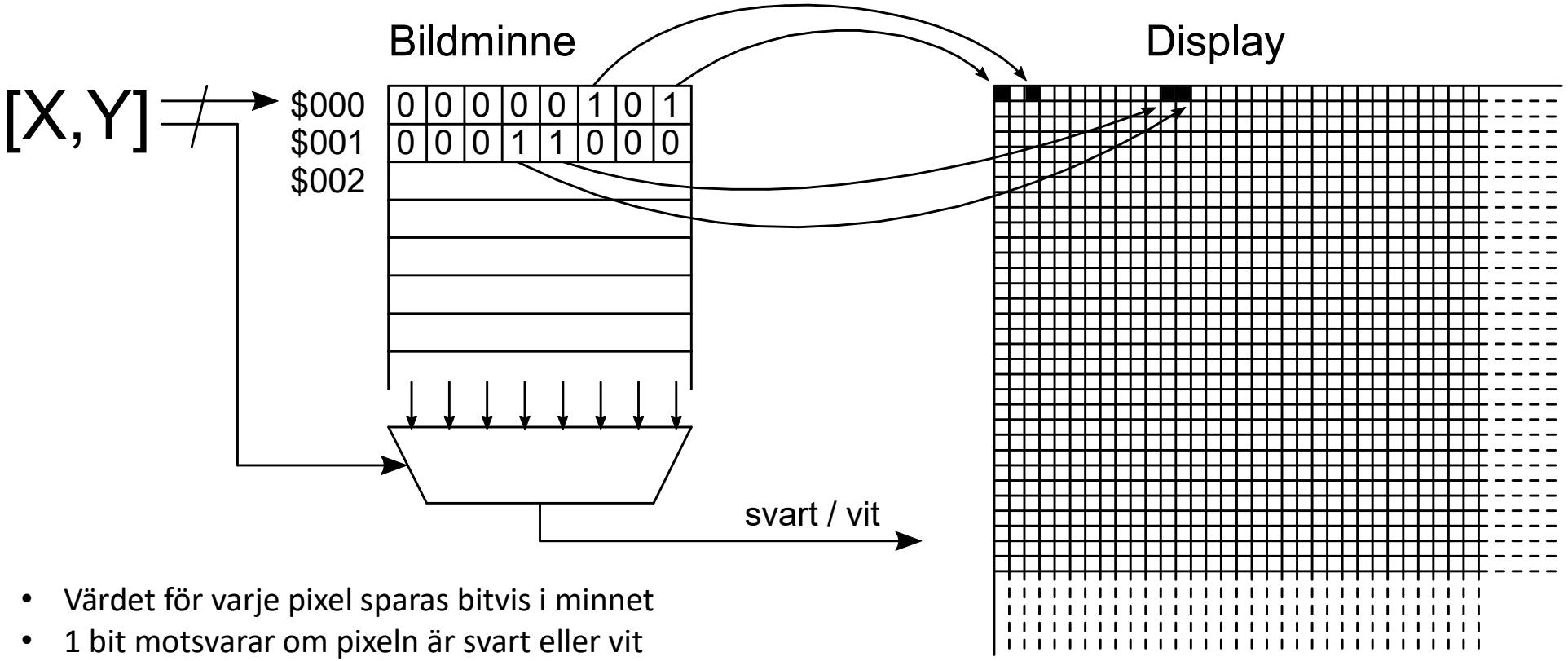


Bildminnesdriven

- CPU:n behöver endast fixa förändringar av bilden
- Fördel : mycket enklare programmering
- Nackdel : dyrare hårdvara

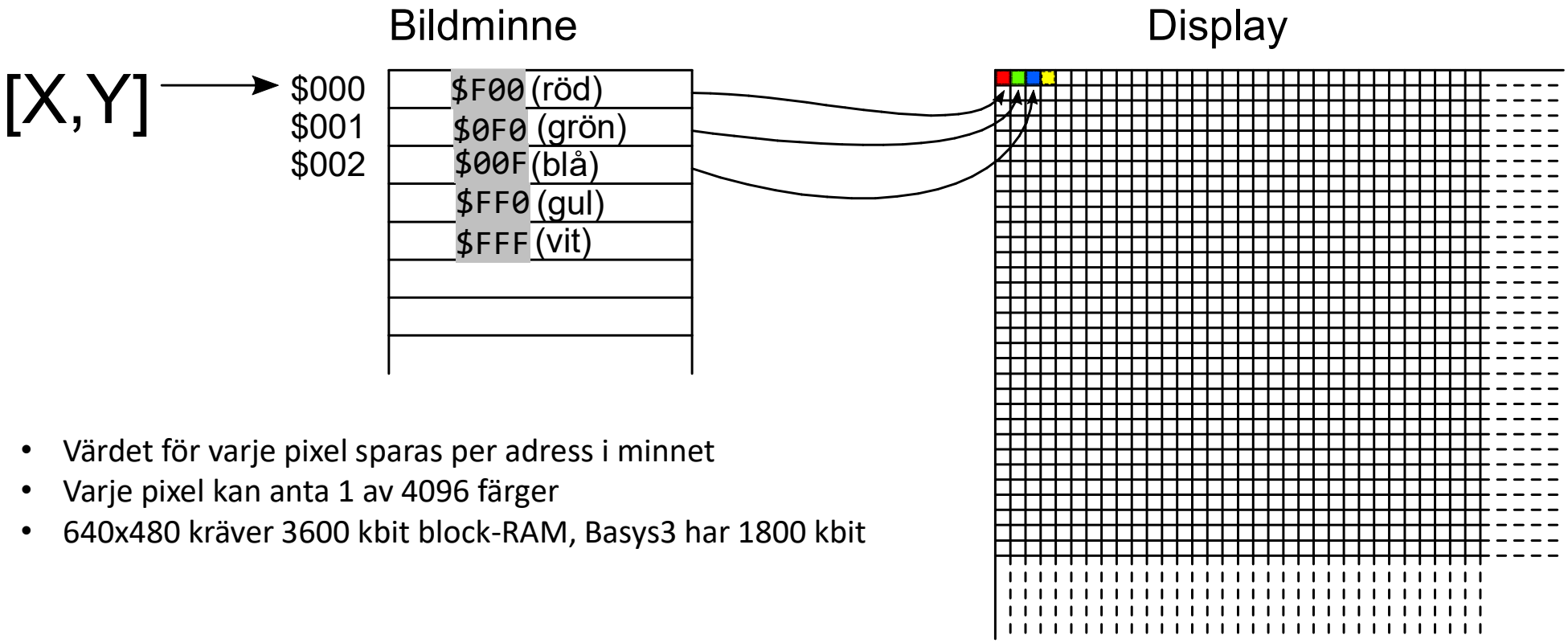


Bitmap-grafik (svart-vit)



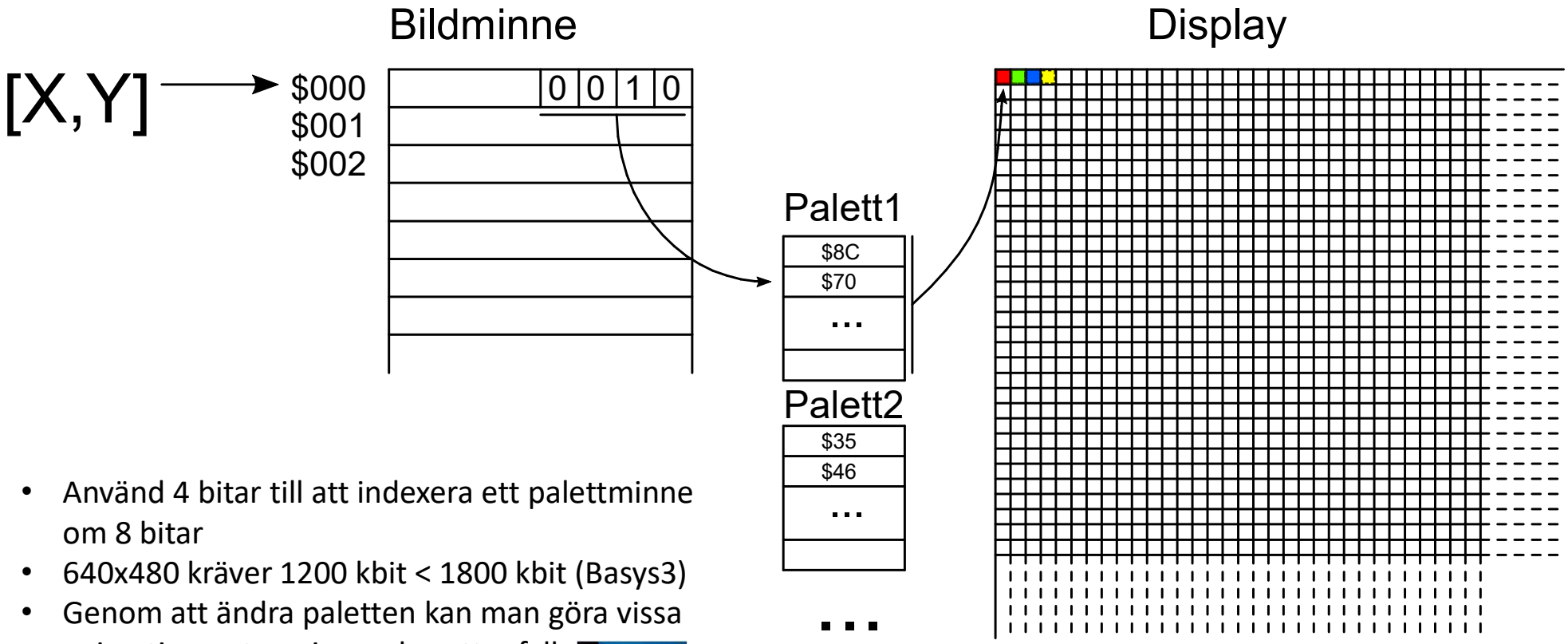
- Värdet för varje pixel sparas bitvis i minnet
- 1 bit motsvarar om pixeln är svart eller vit
- 640x480 kräver 300 kbit block-RAM, Basys3 har 1800 kbit

Bitmap-grafik (12-bitars färg)



- Värdet för varje pixel sparas per adress i minnet
- Varje pixel kan anta 1 av 4096 färger
- 640x480 kräver 3600 kbit block-RAM, Basys3 har 1800 kbit

Bitmap-grafik (4-bitars färg med 8-bitars palett)



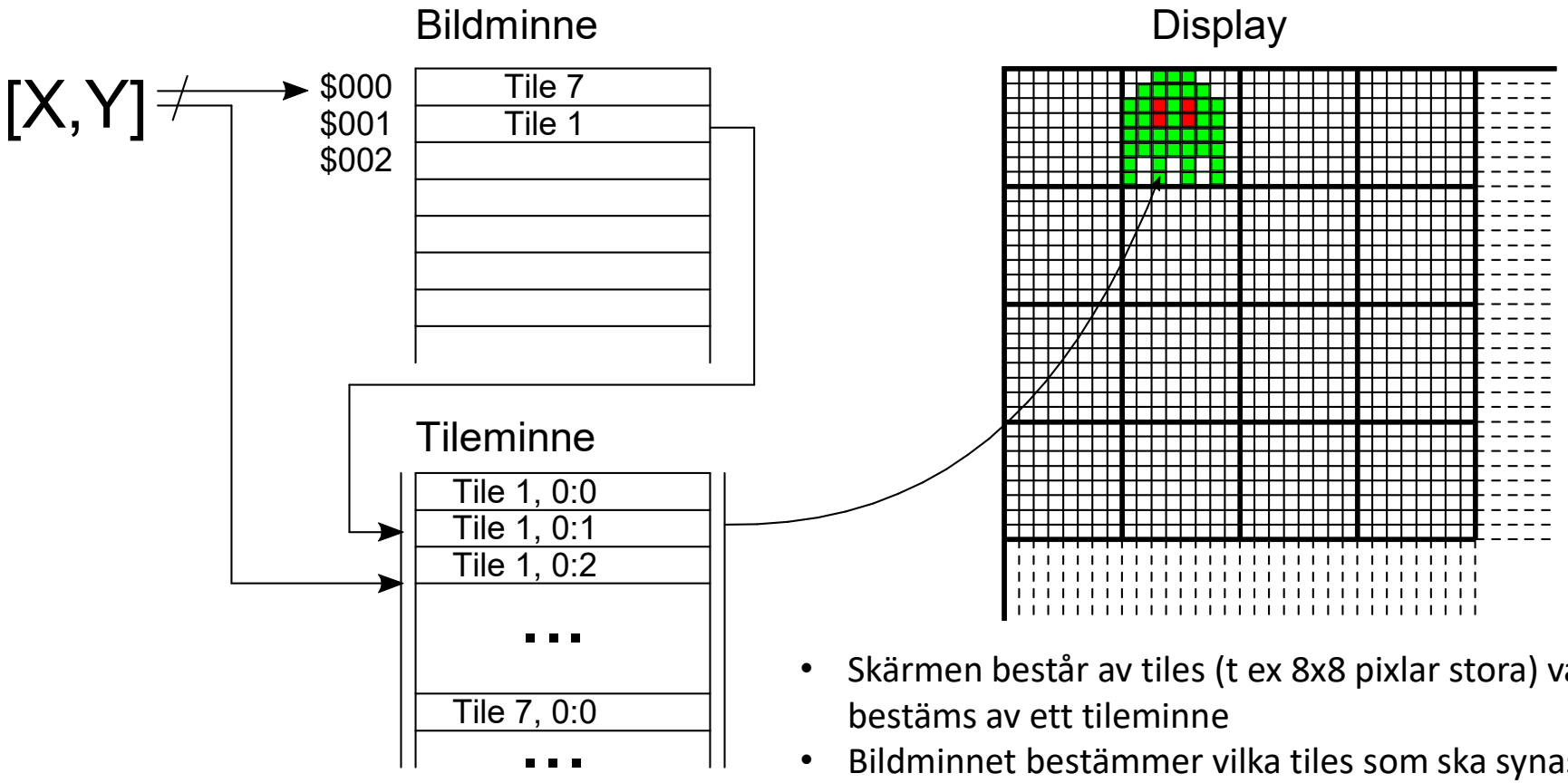
- Använd 4 bitar till att indexera ett palettminne om 8 bitar
- 640x480 kräver 1200 kbit < 1800 kbit (Basys3)
- Genom att ändra paletten kan man göra vissa animationer, t ex rinnande vattenfall

Bitmap-grafik

- Ska man göra riktig 3D-grafik krävs bitmap-grafik, och en snabb CPU förstås
- Bitmap-grafik kräver mycket minne
- Dubbelbuffring är vanligt vid bitmap-grafik, kräver dock dubbla mängden minne
- Alla moderna datorer och spelkonsoller använder bitmap-grafik idag

- Problem 1: FPGA:n på Basys3 har för lite minne för vettig bitmap-grafik
  - Lösning 1: Använd extern RAM (problem för låg bandbredd)
- Problem 2 : Det krävs en mycket snabb CPU för att uppdatera hela bitmap-minnet
  - Lösning 2 : Använd lägre upplösning, t ex 320x200 pixlar
  - Lösning 3 : Använd inte bitmap-grafik

Tile/tecken-baserad grafik



- Skärmen består av tiles (t ex 8x8 pixlar stora) vars utseende bestäms av ett tileminne
- Bildminnet bestämmer vilka tiles som ska synas på skärmen
- Fördel : Kräver ganska lite minne

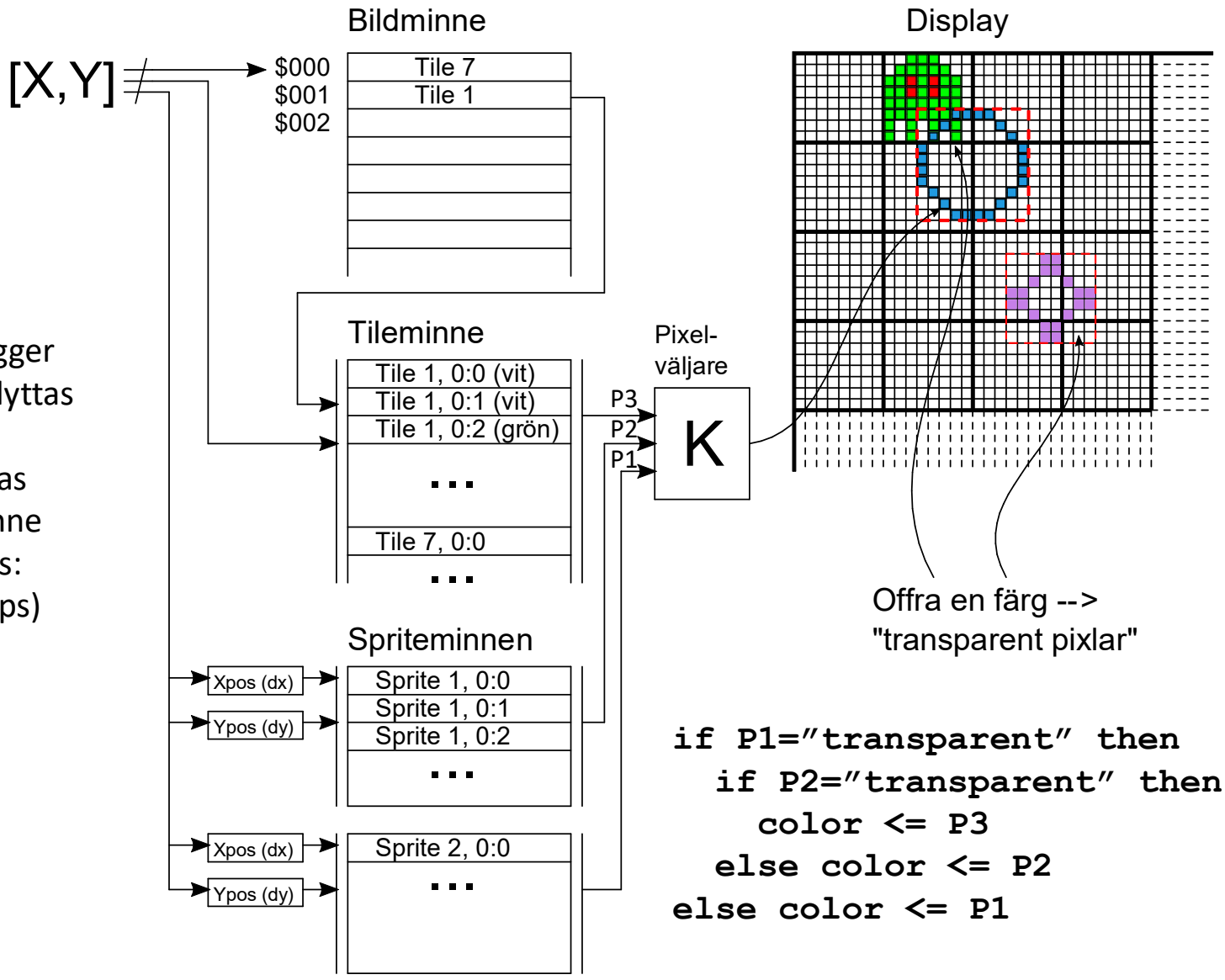
Tile/tecken-baserad grafik

Räkneexempel 8x8 tiles i 256 färger

- En tile är 8x8 pixel och vi vill ha t ex 32 olika tiles
- Med upplösningen 640x480 får vi plats med 80x60 tiles (kräver 4800 bytes bildminnesutrymme)
- Minnesutrymme för tile-utseende : 8x8x32x12 bitar = 24 kbit
- Jämför med bitmap-baserat minne : 640x480 byte = 3600 kbit
- Spel som lämpar sig för tile-grafik : Snake, Tetris, Sokoban
- Trick: Genom att snabbt växla tiles i bildminnet så kan du få till en del roliga effekter varje gång skärmen ritas upp, typ animering.
- Nackdel med tiles : objekt måste röra sig i steg om 8x8 pixlar, lösning → använd sprites

Sprite-baserad grafik

- En sprite är ett objekt som ligger ovanpå spelplanen och kan flyttas i steg om en pixel
- Utseendet för en sprites lagras (lämpligtvis) i ett separat minne
- Lämpliga spel för tiles+sprites:
  -Space invaders (shoot-em-ups)
  -Bilspel
  -Breakout

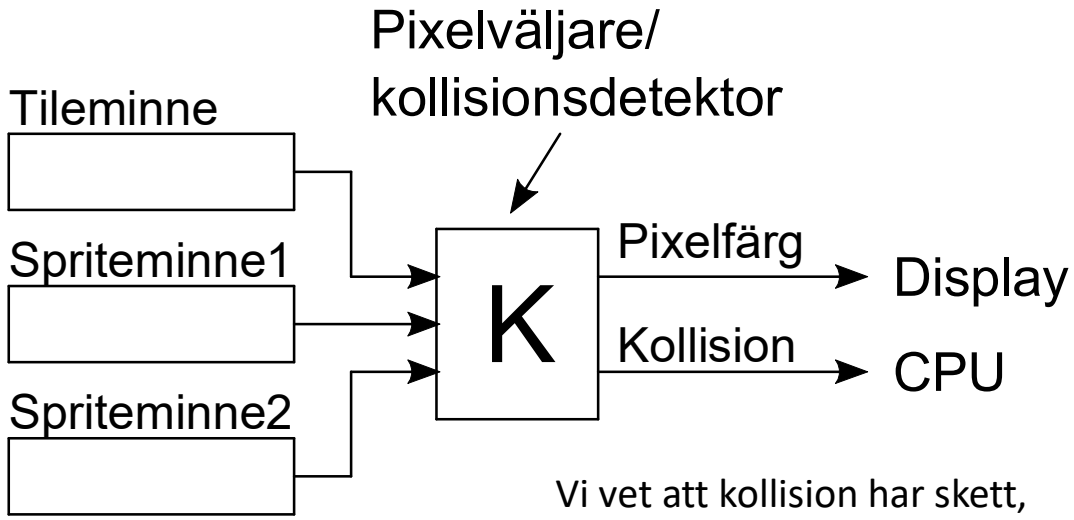



```
if P1="transparent" then
  if P2="transparent" then
    color <= P3
  else color <= P2
else color <= P1
```

Kollisionskontroll

- Alt 1 : Mjukvarukontroll
  Kan kräva mycket beräkningar. Passar bäst för tiles. Svårt för sprites.

- Alt 2 : Hårdvarukontroll →
  När två objekt ska ritas ut samtidigt, alltså på samma pixel, så har dom kolliderat.

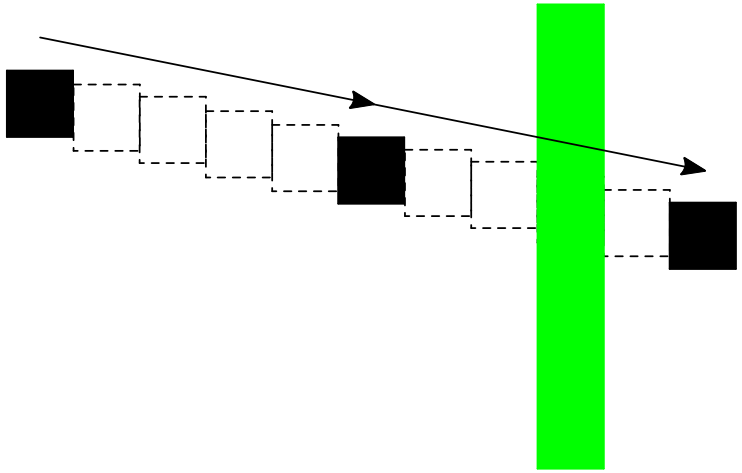


Vi vet att kollision har skett,
men mellan vilka, och hur?
För att svara på det krävs än mer hårdvara.

Rörelse : Hastighet och riktning

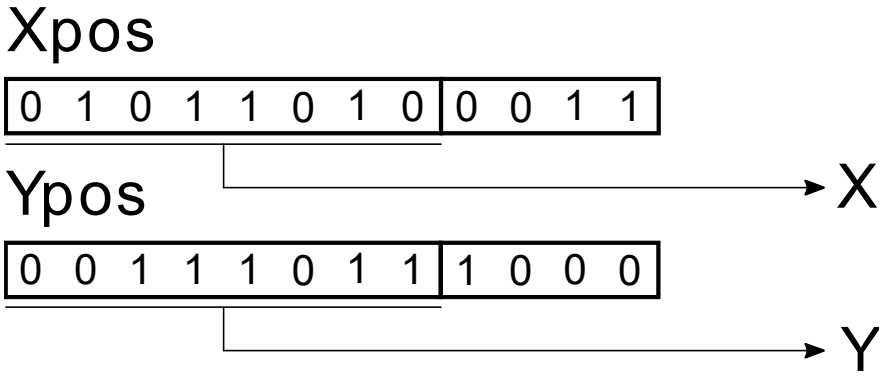
Att flytta ett objekt en pixel per bilduppdatering (60 Hz) tar > 10 s för 640 pixlars bredd. Hur kul är det?

Men vad händer om man ökar hastigheten, dvs flyttar flera pixlar per bilduppdatering?



Lösning ? : Gör alla objekt väldigt tjocka
En generell lösning är svår. Får lösas från fall till fall.

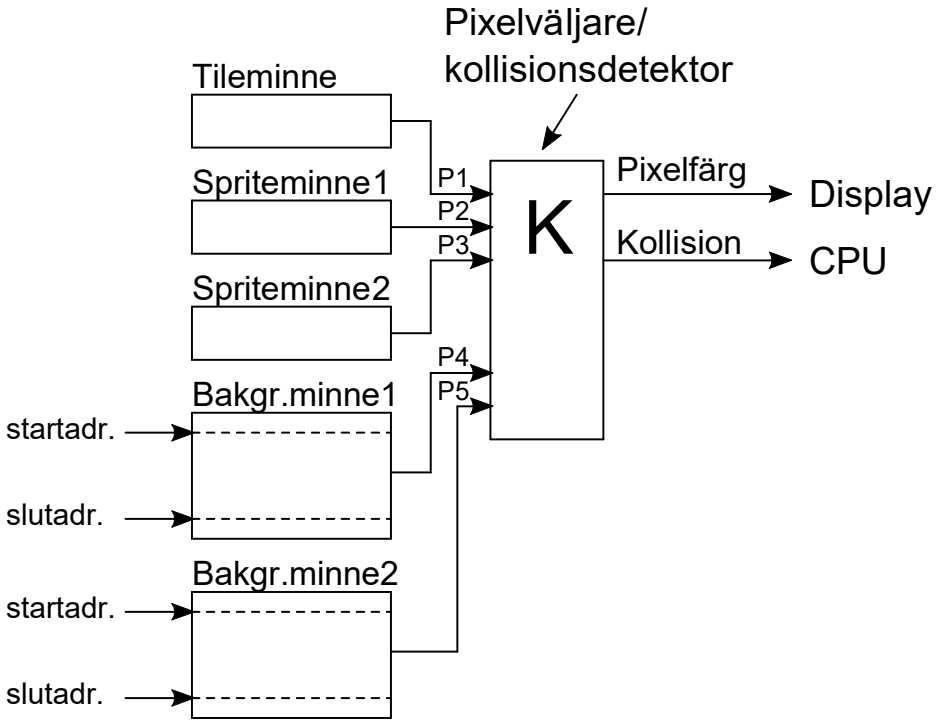
För ”godtyckliga” riktningar, dela upp positionen i en heltalsdel och en decimaldel. (subpixelupplösning) Låt endast heltalsdelen användas vid utritning.



Scrollande bakgrund/förgrund

- Kan läggas till utan att påverka övrig funktionalitet
- Ett relativt enkelt sätt att få visuellt imponerande effekter
- Genom att scrolla flera bakgrunder i olika hastighet kan perspektiv åstadkommas

Genom att flytta start- och slut-pekarna → vid varje bilduppdatering så kommer bakgrundsminnets innehåll att få en visuell scrolleffekt.



Projektgrupp

Bilda projektgrupp

- Projektanmälan senast tisdag 6/2
  -Skapa en grupp
  -Bestäm projekt
  -Indikera hårdvaru-önskemål

- Hitta gruppmedlemmar i kanalen Gruppbildning (Teams)

- Utan grupp, inget projekt!

Projektdemo

Anders Nilsson

www.liu.se

LINKÖPINGS UNIVERSITET