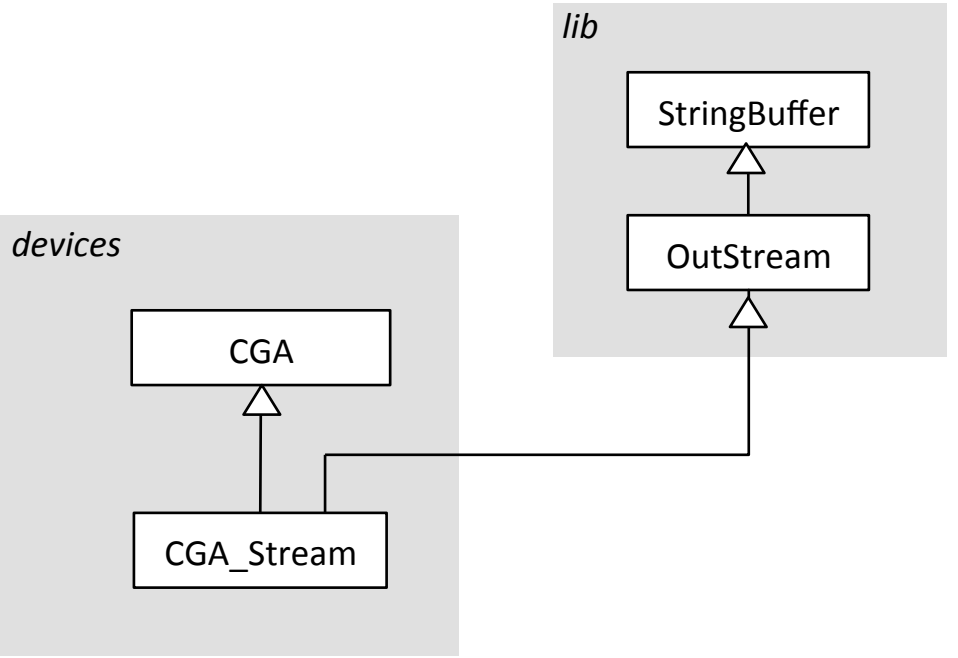
CGA-Programmierung

Ausgabestrom

- StringBuffer: put(c), flush()
  - Sinn der Pufferung? → Performance
  - Sinnvolle Puffergröße? → eine Zeile (80 Zeichen)
- OutStream: ähnlich C++-std::ostream (erweitert StringBuffer)
  - Formatierung, Zahlendarstellung
  - verwendet StringBuffer::put(c)
- CGA_Stream:
  - implementiert flush()
    -> Ausgabe auf Screen
    mithilfe von CGA
- CGA: low-level Zugriff auf Screen



CGA

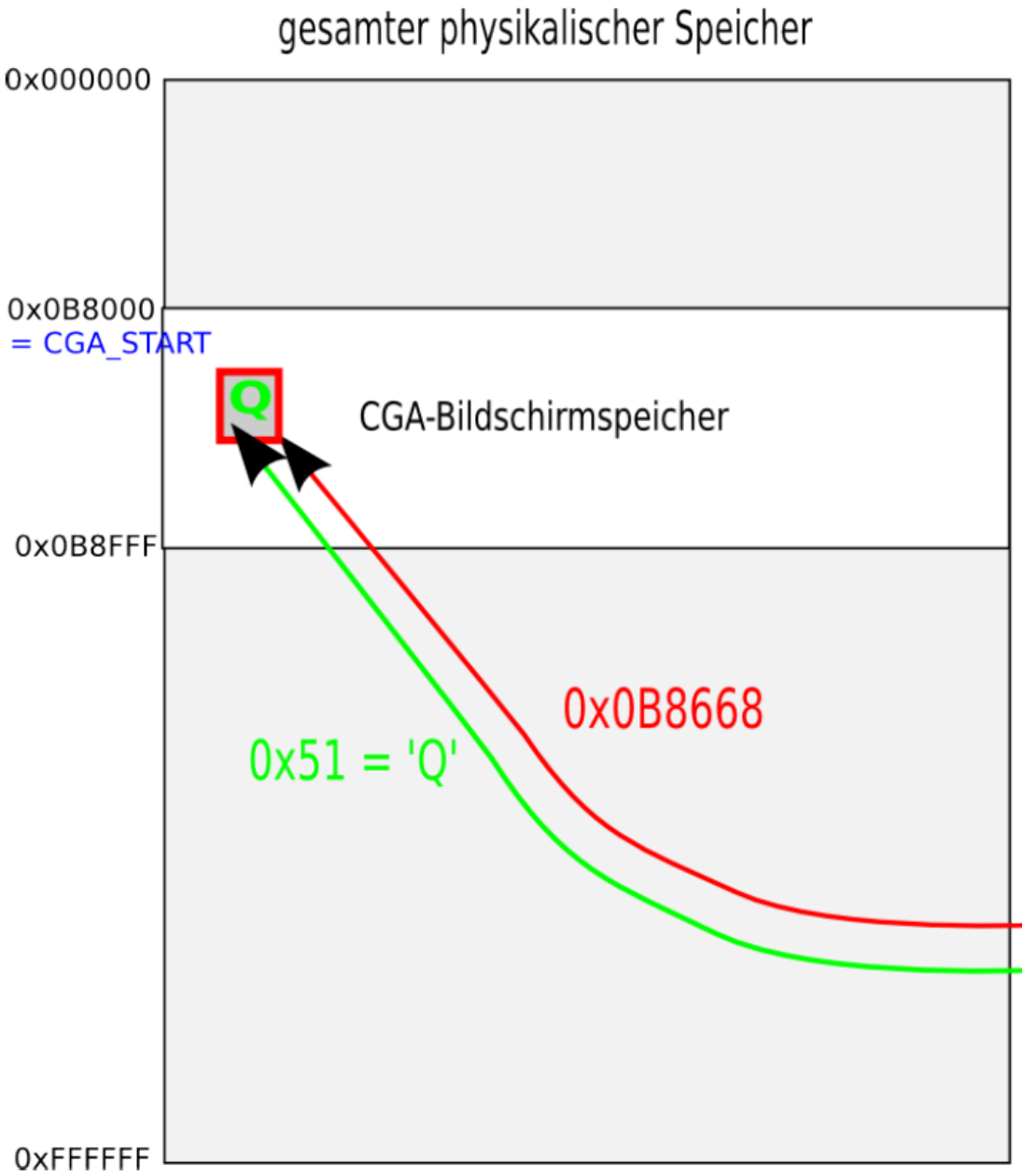
- Wird von CGA_Stream beim flush() verwendet
- `show(x,y,c,attrib)`
  - Zeichen c mit Attribut attrib an Position x/y
  - Code aus dem C++-Crashkurs:

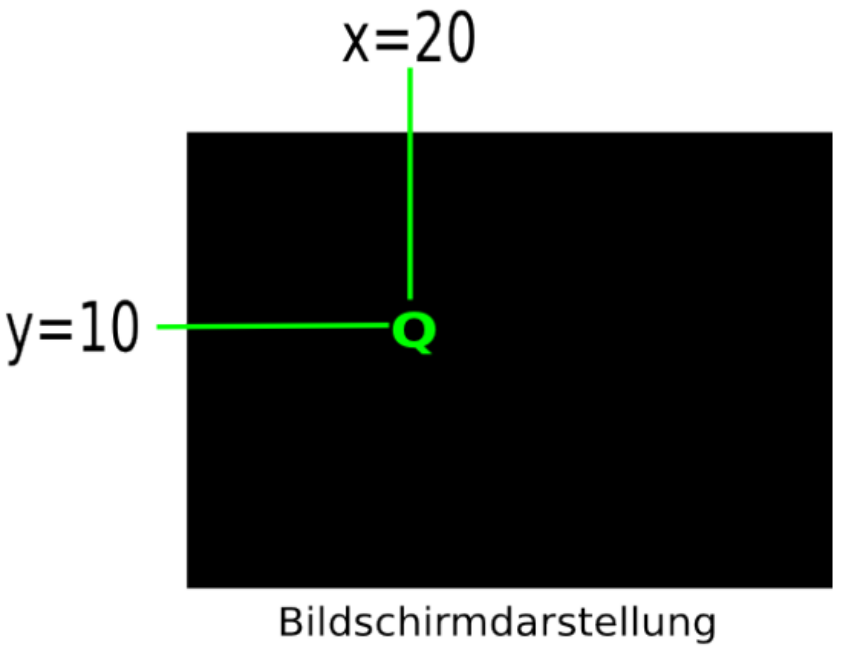
```
char *CGA_START = (char *)0xb8000;
char *pos;
int x=20, y=20;

pos = CGA_START + 2*(x + y*80);
*pos = 'Q';
```

  - Fehlt hier noch etwas?

CGA (2): Standard 80x25 Zeichen






```
#define CGA_START 0xb8000
char *pos;
x=20; y=10;
pos = (char *)CGA_START + 2*(x + y*80);
*pos = 'Q';
```

CGA (3)

- Je zwei Bytes im Bildspeicher pro Bildposition!
- Gerade Adressen: ASCII-Code
- Ungerade Adressen: Attributbyte

```
char *CGA_START = (char *)0xb8000;
char *pos;
int x=20, y=20;

pos = CGA_START + 2*(x + y*80);
*pos = 'Q';
*(pos + 1) = 0x0f; // weiss auf schwarz
```

CGA (4)

- `setpos/getpos`
  - ändern internen Zustand von CGA
  - aktuelle Position wird in `print()` benötigt!
  - positionieren den CGA-Cursor
    (könnte auch abgeschaltet werden, dann irrelevant)

- Allgemein: Zugriff auf PC-Peripherie
  - zwei Adressräume: Speicher-Adressraum, E/A-Adressraum
  - Speicher: direkt über Pointer adressierbar (Graphikspeicher)
  - E/A: über CPU-Befehle in/out (inb/inw/inl; outb/outw/outl)
    - ❖ in HHUos gekapselt in der Klasse `IOport`
  - manche Geräte verwenden sogar beides (z.B. eben CGA)

CGA (5)

- Speziell CGA: Speicher und E/A
  - In Speicher-Adressraum eingeblendeter Graphikspeicher
  - CGA-Register im E/A-Adressraum
- Allgemeines Problem: mehr Register als E/A-Adressen
  → Lösung: Multiplexing über Index-/Datenport

Port	Register	Zugriffsart
3d4	Indexregister	nur schreiben
3d5	Datenregister	lesen und schreiben

Index	Register	Bedeutung
14	Cursor (high)	Zeichenoffset der Cursorposition
15	Cursor (low)	

CGA (6)

- `print(char *text, int length, unsigned char attrib)`
  - Baut auf `show()` und `setpos()` auf
  - Ausgabe am unteren Bildschirmrand angekommen?
    → Scrollen