03-05-H
-709.53

Informatik für Nichtinformatiker (8)

Prof. Dr. Udo Frese
Tobias Hammer

Debugger
Grafik
Animation & Computerspiele

Was bisher geschah

- **Algorithmus: Verfahren zur Berechnung von etwas**
- **Euklidischer Algorithmus: ggT zweier Zahlen**
  - Reduktion: ggT(a, b)=ggT(b, a%b)
  - Initialproblem: ggT(a,0)=a
- **Sieb des Eratosthenes: Liste von Primzahlen**
  - Wenn ein Primzahl gefunden wurde, streiche alle ihre Vielfachen weg
- **Selectionsort bzw. Heapsort: Sortiere eine Liste von Objekten**
  - Suche nach und nach das Minimum der verbliebenen Objekte
- **Dijkstra kürzeste Wege: Finde kürzesten Weg zwischen zwei Knoten in einem Graph**
  - Liste offener und abgeschlossener Knoten
  - schließe jeweils den offenen Knoten mit kürzestem Weg ab
  - wenn der noch nicht abgeschlossen war: füge Nachbarn zu offenen hinzu

Debugger

Debugger

Wie findet man Fehler im Programm?

- **Beim Programmieren geht die Zeit mit Fehler suchen drauf!**
- **Fehler = Bug (vgl. Geschichte der Informatik II)**
- **In kleinen Schritten entwickeln**
- **Problem in Methoden zerlegen (im Übungszettel schon vorgegeben)**
  - Methode programmieren – gründlich testen – dann weitermachen
  - Testfall generieren: Mit einfachen Daten aufrufen
  - Entweder vom Hauptprogramm (außerhalb Methoden/Klassen) aus aufrufen
  - Oder an geeigneter Stelle im Programm (Übung 4: `Map.test`)
- **Methoden schrittweise entwickeln**
  - Methode besteht aus einzelnen Instruktionen
  - Nach jeder Instruktion Ergebnis ausgeben lassen (`puts` oder `pp`) und auf Plausibilität überprüfen
  - Oder: Im Debugger nachverfolgen
- **Beispiel: inifrese0908_erastosthenes.rb**

Debugger

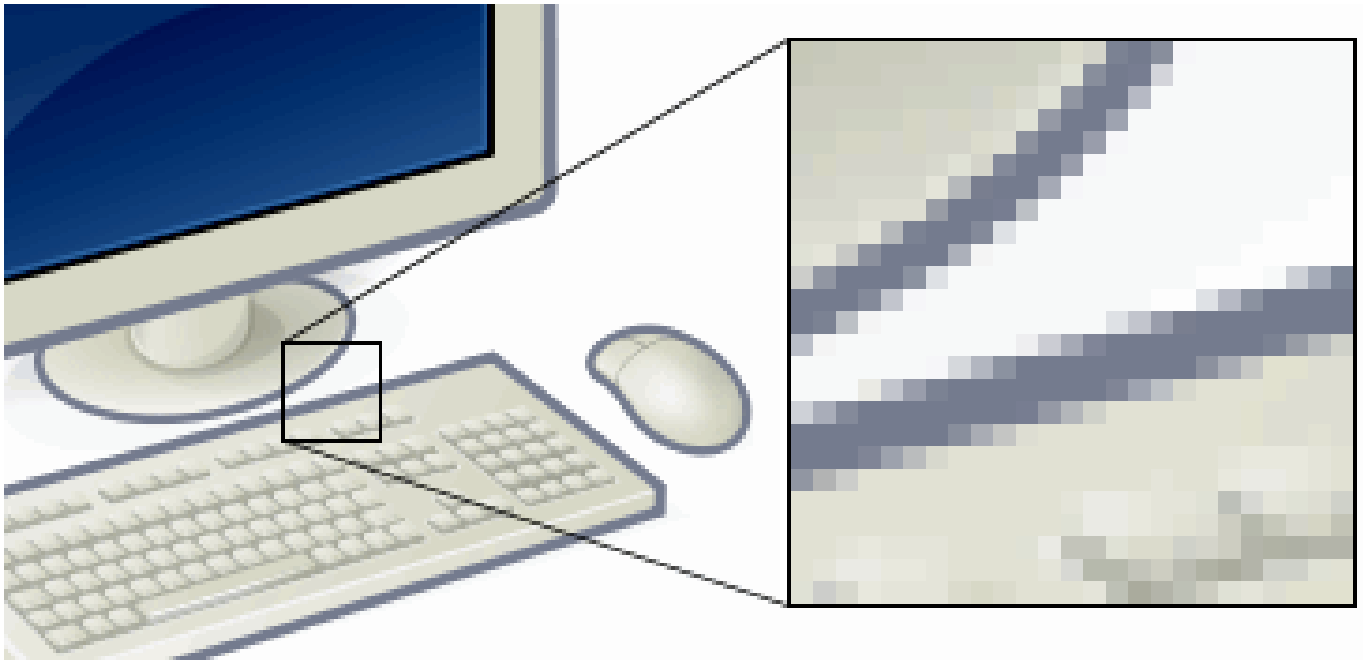
- **Programm zum Fehler finden durch schrittweise Ausführung**
- **Teil von NetBeans/ruby, Menü Debug**
- **Programm im Debugger starten (Debug/Debug main project)**
- **Haltepunkt (Debug/Toggle Line Breakpoint):**
  erreicht das Programm diese Zeile wird die Ausführung gestoppt
- **Einzelnen Schritt ausführen (Debug/Step over, F8)**
- **Einzelnen Schritt ausführen (Debug/Step into, F7), dabei in Methoden hineinspringen, besonders für `.times, .each`**
- **Variablen inspizieren durch Maus über Variablennamen**
- **Variablen im Fenster Variables**
- **Watch: Ausdrücke, die stets angezeigt werden (Debug/New Watch)**
  - Um arrays anzuzeigen: Debug/New Watch `array.pretty_inspect`
- **Fortsetzen (Debug/Continue) oder abbrechen (Debug/Finish)**

Grafik

Grafik

Wie wird ein Bild im Computer dargestellt?

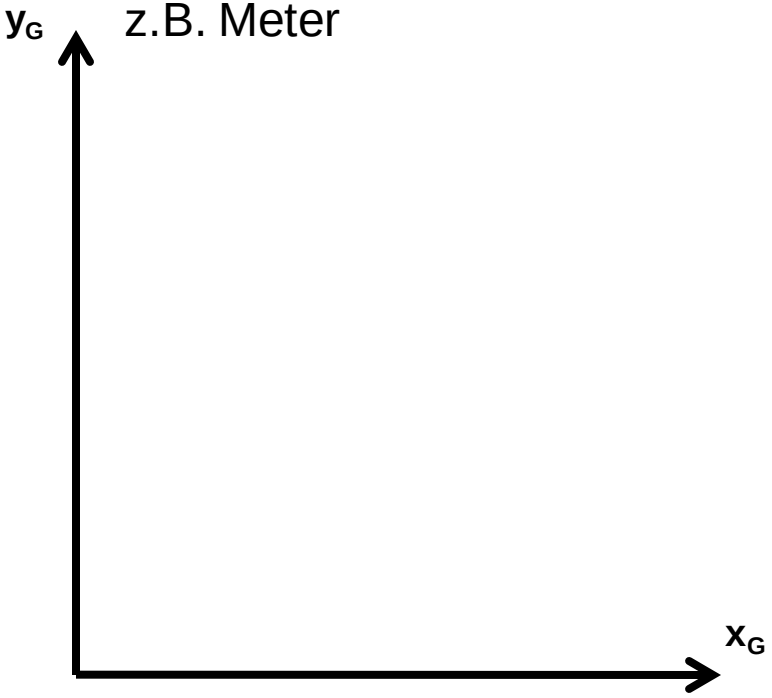
- **Ein zweidimensionales Array von Bildelementen**
- **Raster von Pixeln**
- **Jeder Pixel hat eine Farbe, aber keine innere Struktur**
  ⇒ kleinste Einheit des Bildes



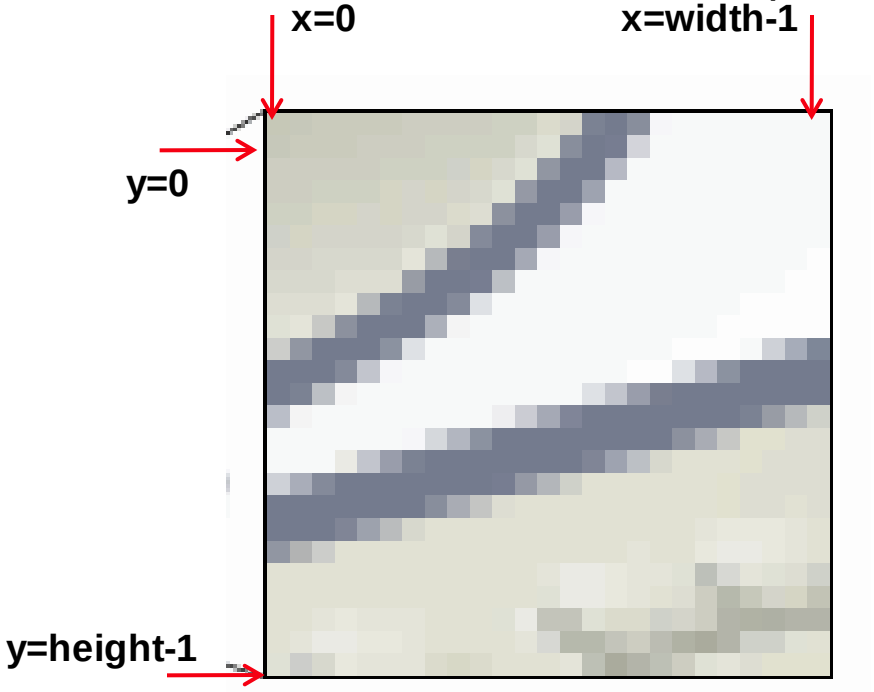
Quelle:
Wikipedia / English / Pixel

Grafik

- **Geometrische Koordinaten**
  - X von links nach rechts
  - Y von unten nach oben
  - Einheit: je nach Anwendung, z.B. Meter

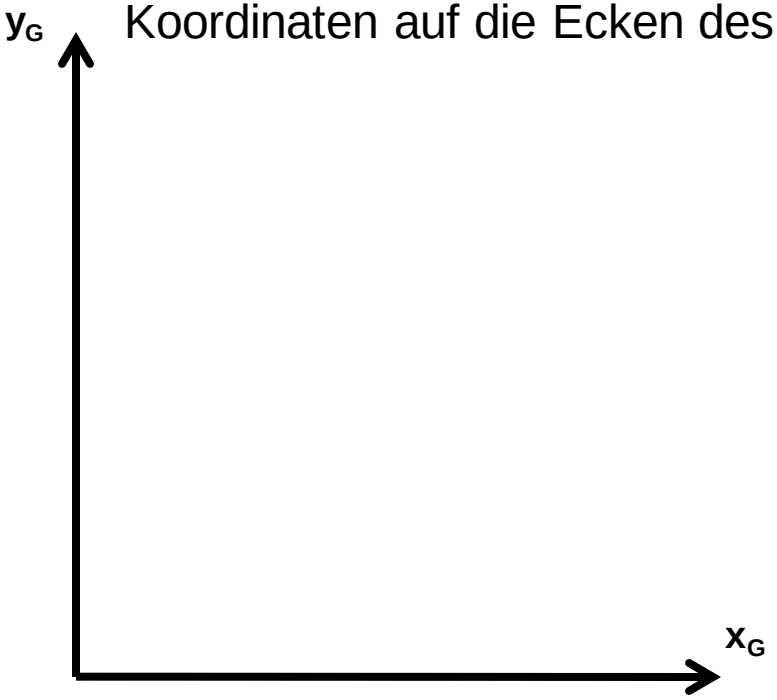


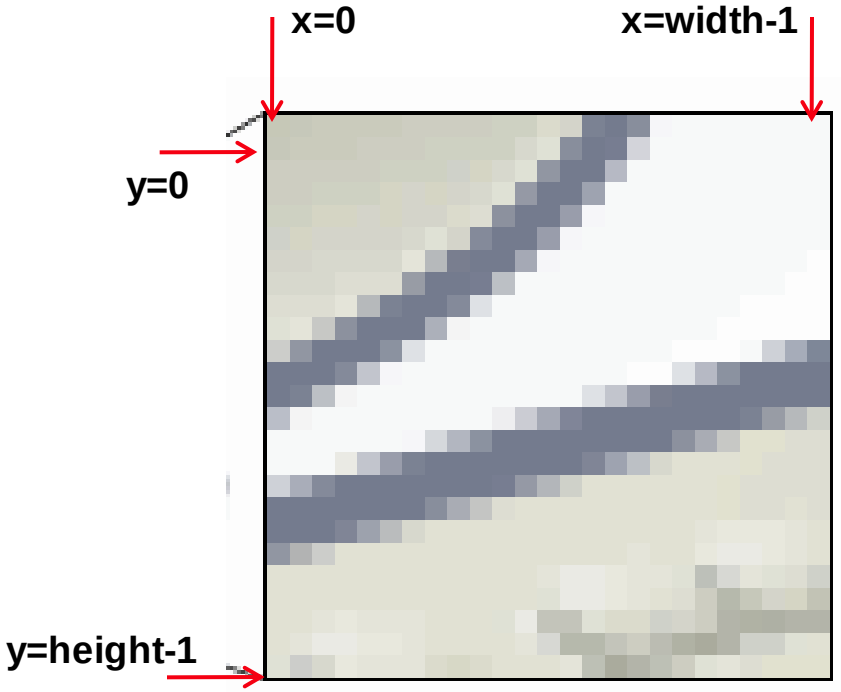
- **Bildkoordinaten im Computer**
  - X von links (0) nach rechts (`width-1`)
  - Y von oben! (0) nach unten (`height-1`)
  - Einheit: fest Pixel
  - Historisch: Schrift →Fernseher→Computer



Grafik

- **Umrechnung mit linearer Funktion**
  - x = (xG-leftG)*width/(rightG-leftG)
  - y = (topG-yG)*height/(topG-bottomG)
  - Bildet leftG, rightG, topG, bottomG in geometrischen Koordinaten auf die Ecken des Bildes ab.





Grafik

Technische Farbdarstellung

- **Aufgabe: Erzeuge (in Grenzen) alle möglichen Farbwahrnehmungen steuerbar mit einem Gerät (Farbfernsehen, Farbfotographie, Farbdruck)**
- **Da der Mensch im Auge drei Farbrezeptoren (Zapfen) hat, durch Überlagerung von drei Primärfarben (Rot, Grün, Blau) unterschiedlicher Helligkeit**
- **Technisch durch**
  - Übereinanderblenden (additiv, alte Beamer, teure DLP Beamer)
  - Feine Punkte nebeneinander (additiv, TV, LCD)
  - Zeitlich hintereinander (additiv, günstige DLP Beamer)
  - Farbpigmente hintereinander (subtraktiv, Farbfotographie, Farbdruck)

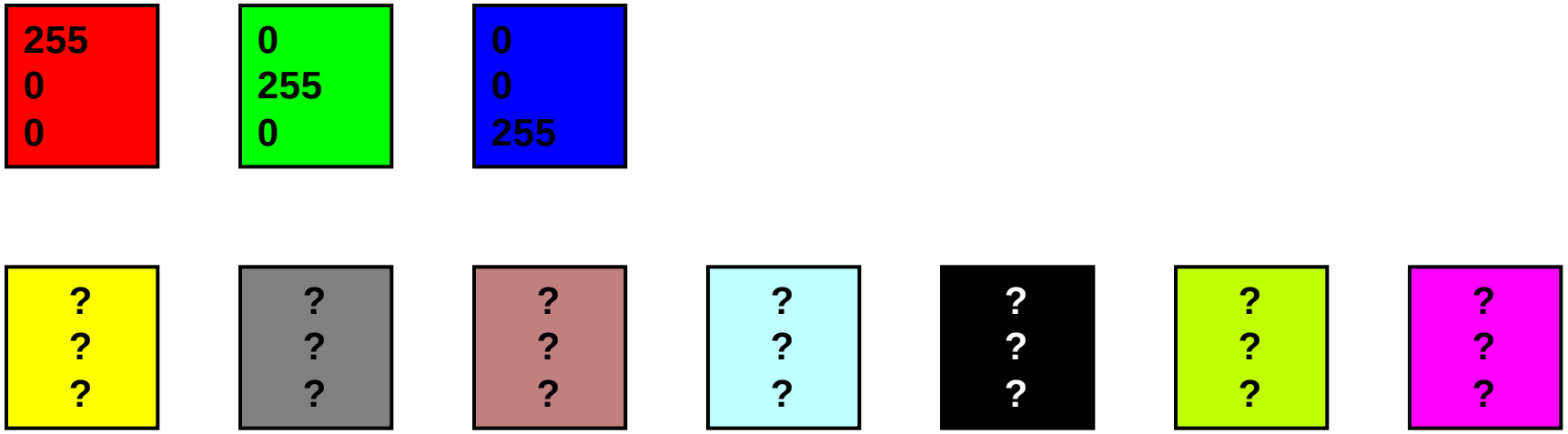
Additiv

Subtraktiv

Grafik

Technische Farbdarstellung

- **Frage an das Auditorium: Wie mischt man die folgenden Farben additiv aus Rot [0..255], Grün [0..255] und Blau [0..255]?**



Grafik

Technische Farbdarstellung

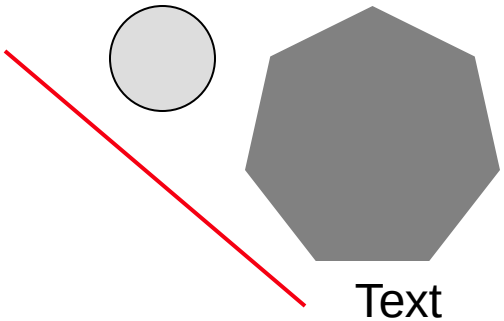
- Frage an das Auditorium: Wie mischt man die folgenden Farben additiv aus Rot [0..255], Grün [0..255] und Blau [0..255]?

255 0 0	0 255 0	0 0 255				
255 255 0	128 128 128	192 128 128	192 192 255	0 0 0	192 255 0	255 0 255

Grafik

Geometrische Grundelemente

- **Programmierer betrachtet selten einzelne Pixel**
- **Sondern zeichnet geometrische Grundelemente**
  - Linien
  - Kreise (Umriss oder gefüllt)
  - gefüllte Rechtecke
  - gefüllte Polygone
  - Bilder (aus .jpg/.png Datei)
  - Schrift
- **Jedes Grundelement erfordert einen speziellen Algorithmus um es in Pixel zu übersetzen („rendern“)**
  - sehr rechenzeitkritisch
  - Aufgabe einer speziellen Grafikbibliothek (Klassensammlung)



Grafik

Grafik mit Rubygame

- Eine Farbe bzw. ein Pixel ist Array `[rot, grün, blau]` jeweils 0..255
- Bild aber *nicht* Array von Arrays wie in `Array.new(width, Arraw.new(height, [255, 255, 255]))`
- sondern mit spezieller Bibliothek (Klassensammlung) Rubygame, die schnelles und komfortables Arbeiten ermöglicht
- Keine Bibliothek für Fenster, Menüs, Buttons, etc.
- Sondern dafür direkt etwas zu malen und das Ergebnis zu sehen
- Hier ein Überblick über was wir brauchen
- `http://rubygame.org/docs/rdoc/2.3.0/`

Grafik

Grafik mit Rubygame

- Punkte jeweils als Array `[x, y]`
- Farben jeweils als Array `[r,g,b]`
- Fenster öffnen:
  `screen=Rubygame::Screen.open([width, height])`
- Linie: `screen.draw_line(from, to, color)`
- Kreis (Umriss bzw. ausgefüllt)
  `screen.draw_circle(center, radius, color)`
  `screen.draw_circle_s(center, radius, color)`
- Rechteck (gefüllt):
  `screen.draw_box_s(left_top, right_bottom, color)`
- Polygon (gefüllt):
  `screen.draw_polygon_s(points, color)`,
  `points` Array von Punkten

Grafik

Grafik mit Rubygame

- **Bilder**
  - Laden: `image = Rubygame::Surface.load ("image.png")`
  - Malen: `image.blit (screen, left_top)`
  - Blit steht für "block image transfer"
- **Schrift ist leider kompliziert**
  - Schriftartensystem initialisieren mit `Rubygame::TTF.setup`
  - Schriftart laden: `font=Rubygame::TTF.new("FreeSans.ttf", size)`
  - Datei FreeSans.ttf muss im Verzeichnis sein (frei kopierbar)
  - Text rendern: `image = font.render(text, true, color)`
  - Ergebnis ins Bild kopieren: `image.blit (screen, left_top)`
- **Ergebnis sichtbar machen: `screen.update`**
- **Am Ende: `Rubygame.quit`**

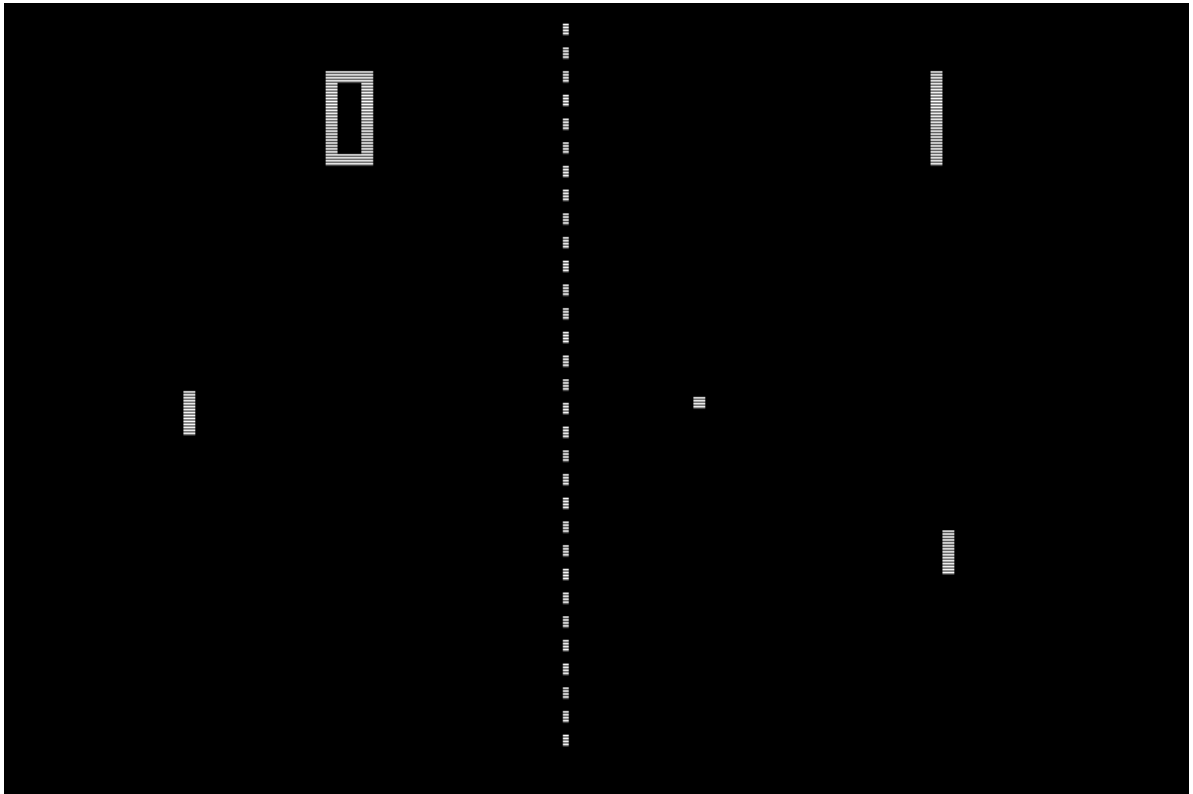
Animation und Computerspiele

- **Allgemeine Übersicht, unter:**
  http://de.wikipedia.org/wiki/Computerspiel
- **Empfehlung:**
  http://www.worldofgoo.com/

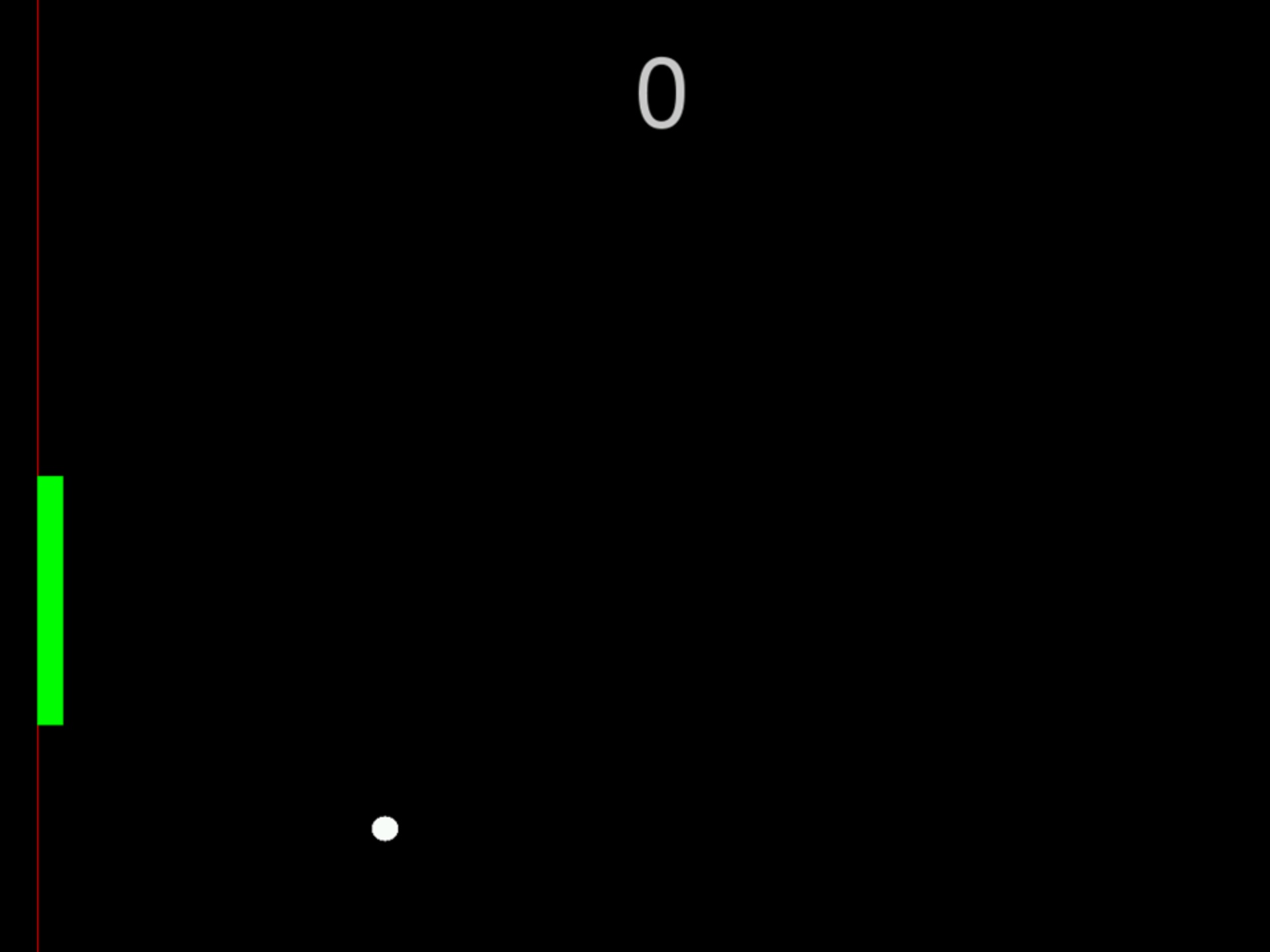
Animation und Computerspiele

Beispiel: Pong

- **Quele: http://www.youtube.com/watch?v=LPkUvfL8T1I**
- **Urahn aller Videospiele (1972), Atari**

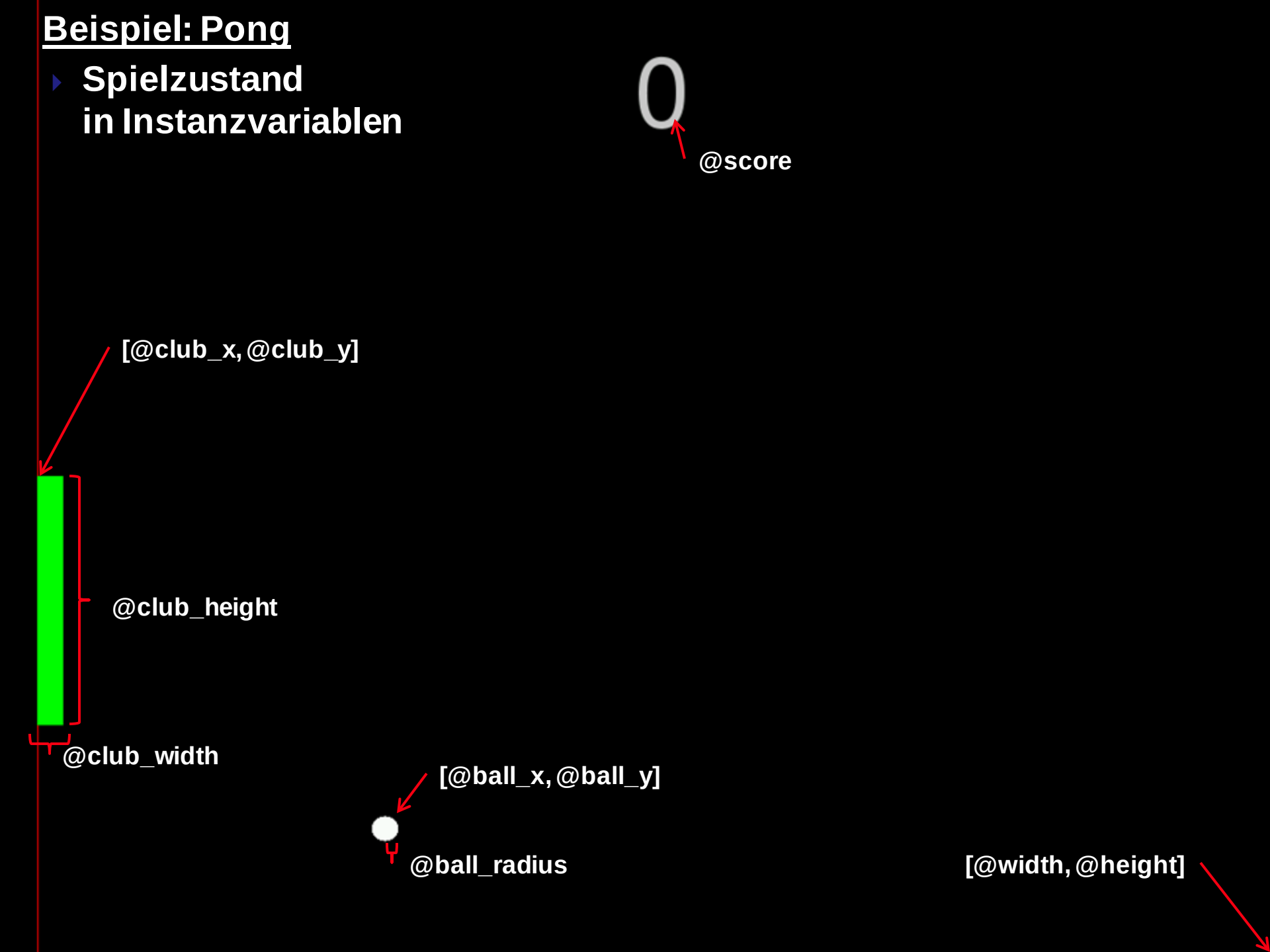


Video 1


0

Beispiel: Pong

- Spielzustand
  in Instanzvariablen

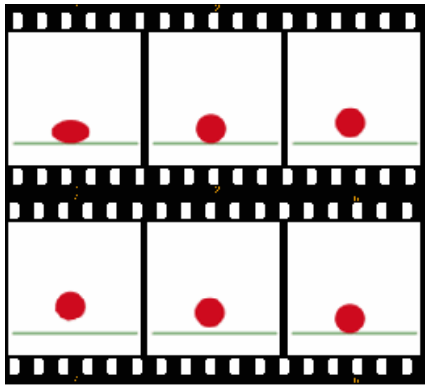


Animation und Computerspiele

- **Beispiel: Pong, RubyGameExample/main.rb**
- **Instanzvariable für Rubygame: `@screen, @font`**
- **Instanzvariablen für Spielzustand:**
  - `@ball_x, @ball_y, @ball_radius,`
  - `@club_x, @club_y, @club_width, @club_height,`
  - `@score`
- **Methode: `initialize`, `draw, new_game`**

Animation und Computerspiele

- **Animation (von lat. *animare*, „zum Leben erwecken“) ist im engeren Sinne jede Technik, bei der durch das Erstellen und Anzeigen von Einzelbildern für den Betrachter ein bewegtes Bild geschaffen wird.**
  Quelle: wikipedia.de/animation
- **Begriff wird nicht für abgefilmte Bewegungen verwendet**
- **>12 Bilder /s nötig**



Animation und Computerspiele

Grundstruktur aller Computerspiele (1)

- **Definiere Spielzustand (Position, Geschwindigkeit, Punkte, etc.)**
- **Berechnungen in endloser Schleife, mehrfach pro Sekunde**
- **main-loop-Architektur**
- **Wiederhole bis Abbruch**
  - Rechne Spielzustand einen Schritt weiter
  - Zeichne aktuellen Spielzustand in Bild
  - Zeige Bild auf dem Bildschirm an
  - Warte bis Zykluszeit (1/framerate) vergangen

```
def initialize
    …
    @clock = Rubygame::Clock.new
    @clock.target_framerate = 24
end

def loop
  while true do
    update
    draw
    @screen.update
    @clock.tick
  end
  Rubygame.quit
end
```

Animation und Computerspiele

Beispiel: Pong, RubyGameExample/main.rb

- **Instanzvariablen für Bewegung: `@ball_vx, @ball_vy, @club_vy`**
- **Methoden: `loop`, `update`, `update_ball`**

Animation und Computerspiele

- **„Computerspiele sind interaktive Animationen“, d.h. sie reagieren auf den Spieler**
  - Joystick
  - Mausklick
  - Tastatur

Animation und Computerspiele

Grundstruktur aller Computerspiele (2)

- **Definiere Spielzustand (Position, Geschwindigkeit, Punkte, etc.)**
- **Wiederhole bis Abbruch**
  - Werte Eingabeereignisse aus (ggf. Abbruch)
  - Rechne Spielzustand einen Schritt weiter
  - Zeichne aktuellen Spielzustand in Bild
  - Zeige Bild auf dem Bildschirm an
  - Warte bis Zykluszeit (1/framerate) vergangen

```
def initialize
    @events =
   Rubygame::EventQueue.new
    @clock = Rubygame::Clock.new
    @clock.target_framerate = 24
end
def loop
  while true do
    if handle_events==:quit
   then break end
    update
    draw
    @screen.update
    @clock.tick
  end
  Rubygame.quit
end
```

Animation und Computerspiele

Eingabeereignisse in Rubygame

- **Alle Ereignisse durchlaufen mit: `events.each do |event| … end`**
- **Typ abfragen mit `if event.is_a?(Eventklasse) then`**
- **`Rubygame::QuitEvent`**
  - Spieler hat das Fenster geschlossen
- **`Rubygame::JoyAxisEvent`, `Rubygame::JoyBallEvent`**
  - Joystick (betrachten wir nicht)
- **`Rubygame::KeyDownEvent`, `RubyGame::KeyUpEvent`**
  - `event.key` is gedrückte/losgelassene Taste (`Rubygame::K_A,…,K_Z, K_UP, K_DOWN,K_ESCAPE, …`)
- **Taste, nicht Zeichen (`Rubygame::K_A!=„A“`)**
- **`Rubygame::MouseDownEvent`, `Rubygame::MouseUpEvent`, `Rubygame::MouseMotionEvent`**
  - Mausbewegung und klicken

Animation und Computerspiele

Beispiel: Pong, RubyGameExample/main.rb

- **Methode: `handle_events, update_club`**

Zusammenfassung

- **Fehler im Programm finden**
  - Schritt für Schritt entwickeln, jeden Schritt testen, Kontrollausgaben
  - Debugger für schrittweise Ausführung verwenden
- **Grafik**
  - Bilder im Computer als 2D Array von Farbwerten (Pixel)
  - Rendern von geometrischen Grundelementen mit Grafikbibliothek (für uns: Rubygame)
- **Animation und Computerspiele**
- **Illusion der Bewegung durch schnelle Folge von Einzelbildern**
- **Endlos wiederholte Hauptschleife (main-loop-Architektur):**
  - Ereignisse holen
  - Zustand weiterrechnen
  - zeichnen
  - anzeigen
  - warten