252-0027

Einführung in die Programmierung

Übungen

PS1 Nachbesprechung & Hoare Logic

Henrik Pätzold
Departement Informatik
ETH Zürich

Heutiger Plan

- RequestFeedback.txt
- Nachbesprechung Probeprüfung
- Hoare Logic
- Weakest Precondition

Aufgabe a)

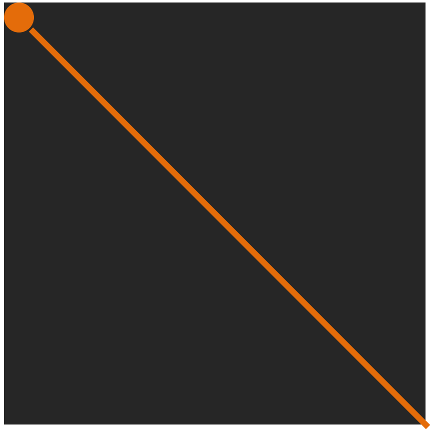
Implementieren Sie die Methoden `fillRectangle` und `drawDiagonal` der Klasse `ImageFrame`. Die Dimensionen eines ImageFrames ist gegeben durch das Pixel-Array `frame`. Sie dürfen davon ausgehen, dass es sich dabei immer um eine rechteckige Matrix handelt, welche mit nicht-null Pixeln gefüllt ist.

- `fillRectangle(int x0, int y0, int x1, int y1, int r, int g, int b)`:
  Diese Methode soll das Rechteck von (x0,y0) bis und mit (x1,y1) mit der Farbe (r,g,b) einfärben.

- `drawDiagonal(int x0, int y0, int r, int g, int b)`:
  Diese Methode soll eine Diagonale mit Startpunkt (x0,y0) und Farbe (r,g,b) zeichnen. Das heisst die Pixel an Position (x0,y0), (x0+1, y0+1), (x0+2, y0+2), ... sollen eingefärbt werden, so dass die Diagonale bis an den Rand des Pixel-Arrays reicht.

Sie dürfen davon ausgehen, dass die Methode nur mit x- und y-Werten aufgerufen wird, welche innerhalb der Matrix liegen und r,g und b Werte von 0 bis 255 haben.

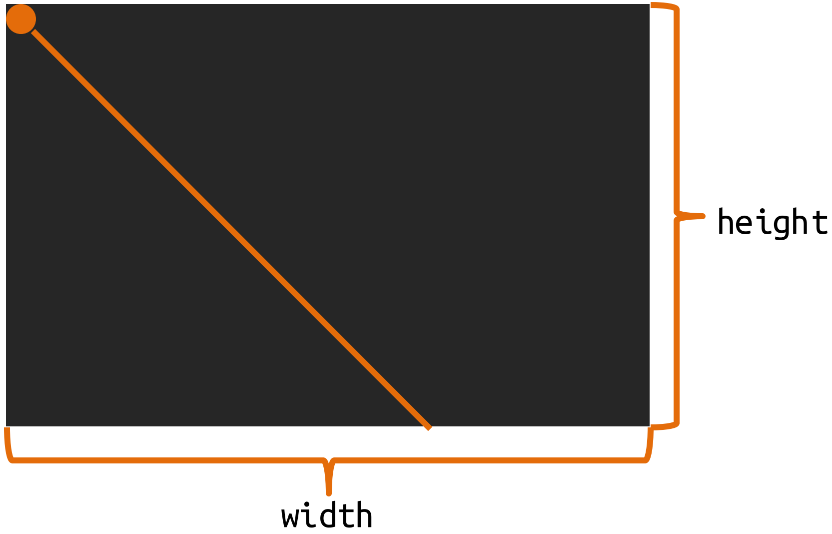
Nachbesprechung – Malen einer Diagonale



```java
public static void drawDiagonal(int x0, int y0, int r, int g, int b){
    int min = Math.min(frame.height, frame.width);
    for(int i = 0; i < min - x0; i++){
        frame[x0+i][y0+i] = new Pixel(r,g,b);
    }
}
```

x ist für die Horizontale,
y für die vertikale

Nachbesprechung – Malen einer Diagonale

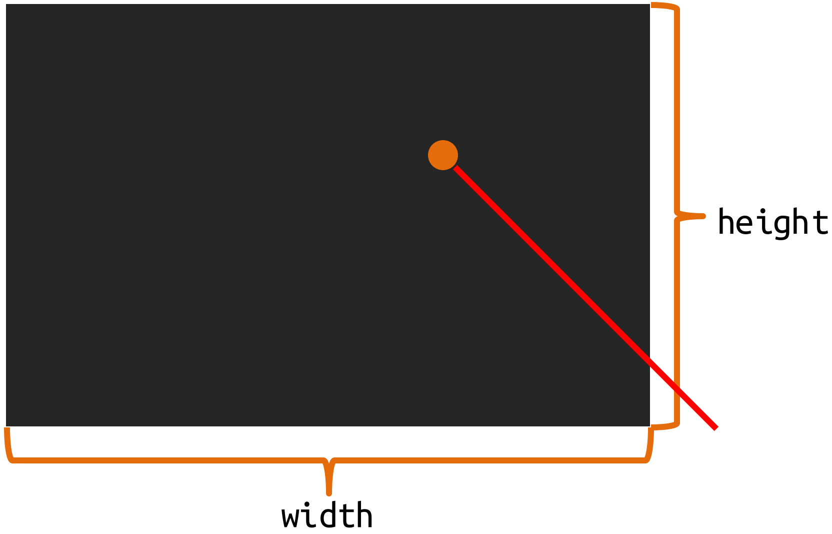



```java
public static void drawDiagonal(int x0, int y0, int r, int g, int b){
    int min = Math.min(frame.height, frame.width);
    for(int i = 0; i < min - x0; i++){
        frame[x0+i][y0+i] = new Pixel(r,g,b);
    }
}
```

x0+i < min = height

x ist für die Horizontale,
y für die vertikale

Nachbesprechung – Malen einer Diagonale




```java
public static void drawDiagonal(int x0, int y0, int r, int g, int b){
    int min = Math.min(frame.height, frame.width);
    for(int i = 0; i < min - x0; i++){
        frame[x0+i][y0+i] = new Pixel(r,g,b);
    }
}
```

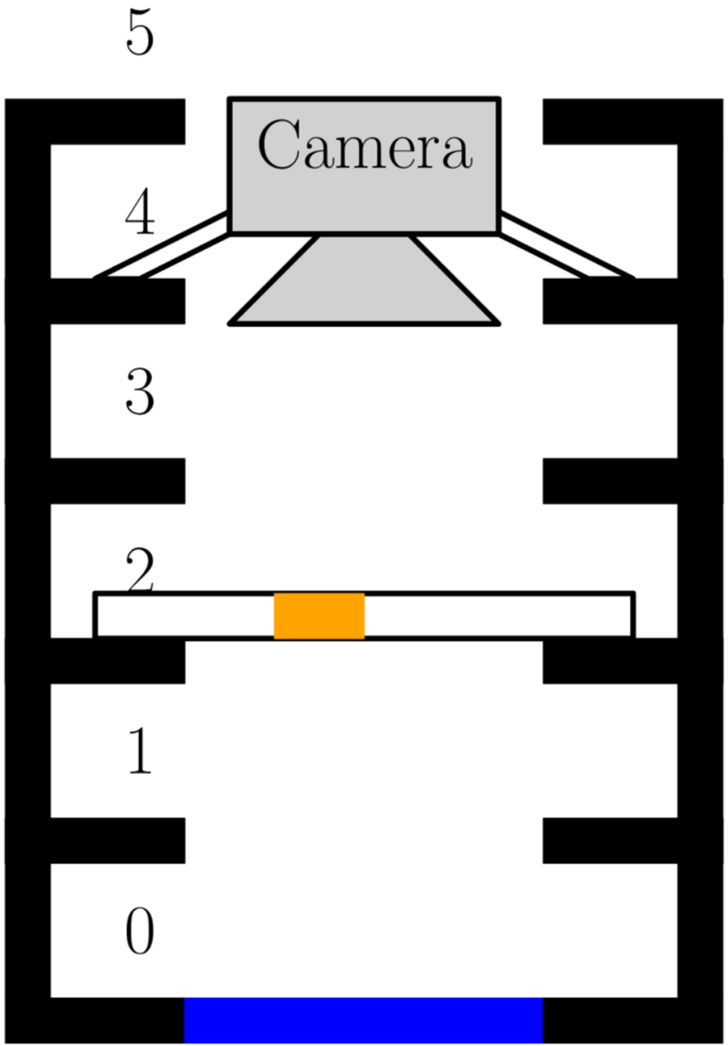
x0+i < min = height

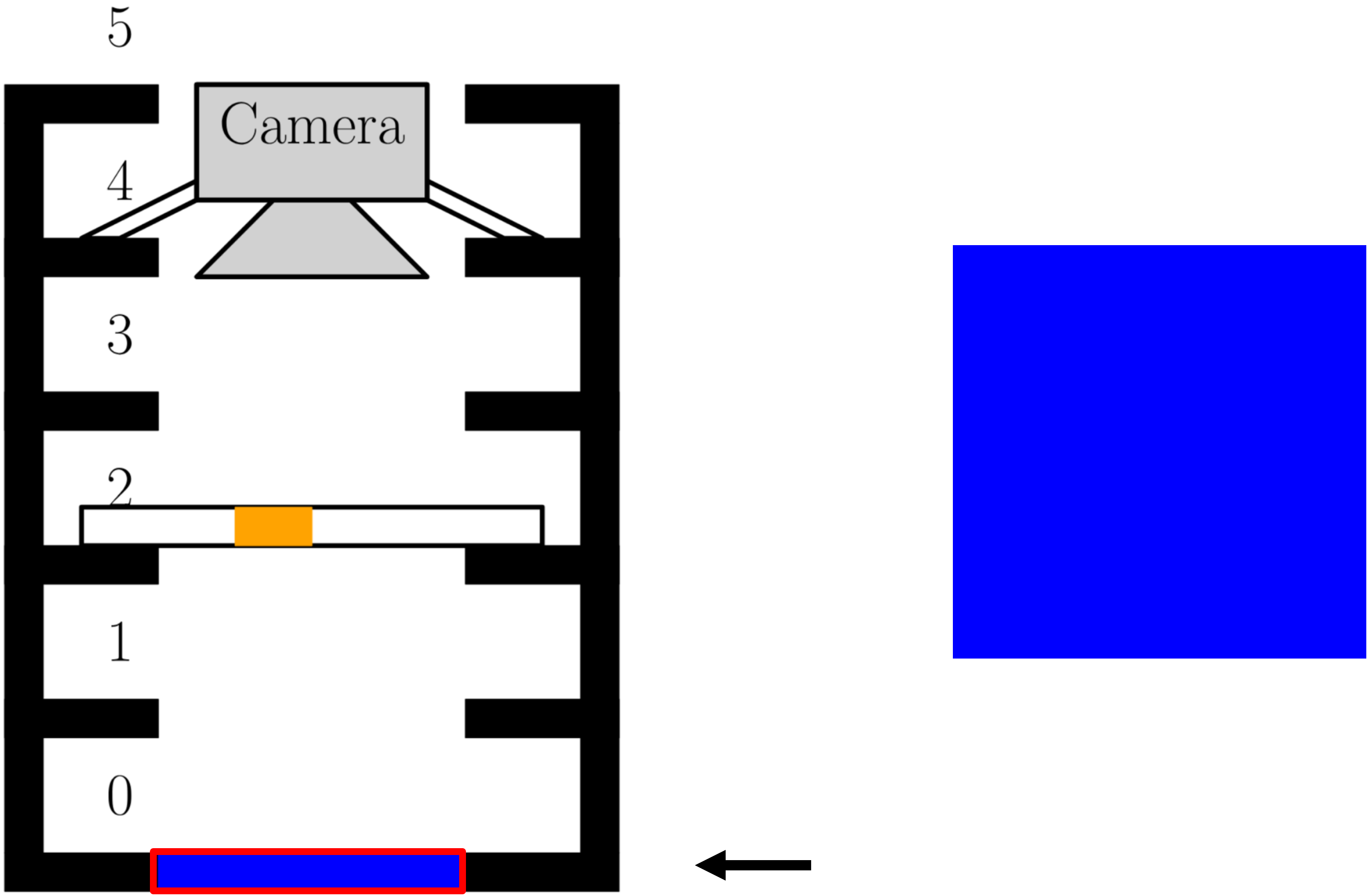
x ist für die Horizontale,
y für die vertikale

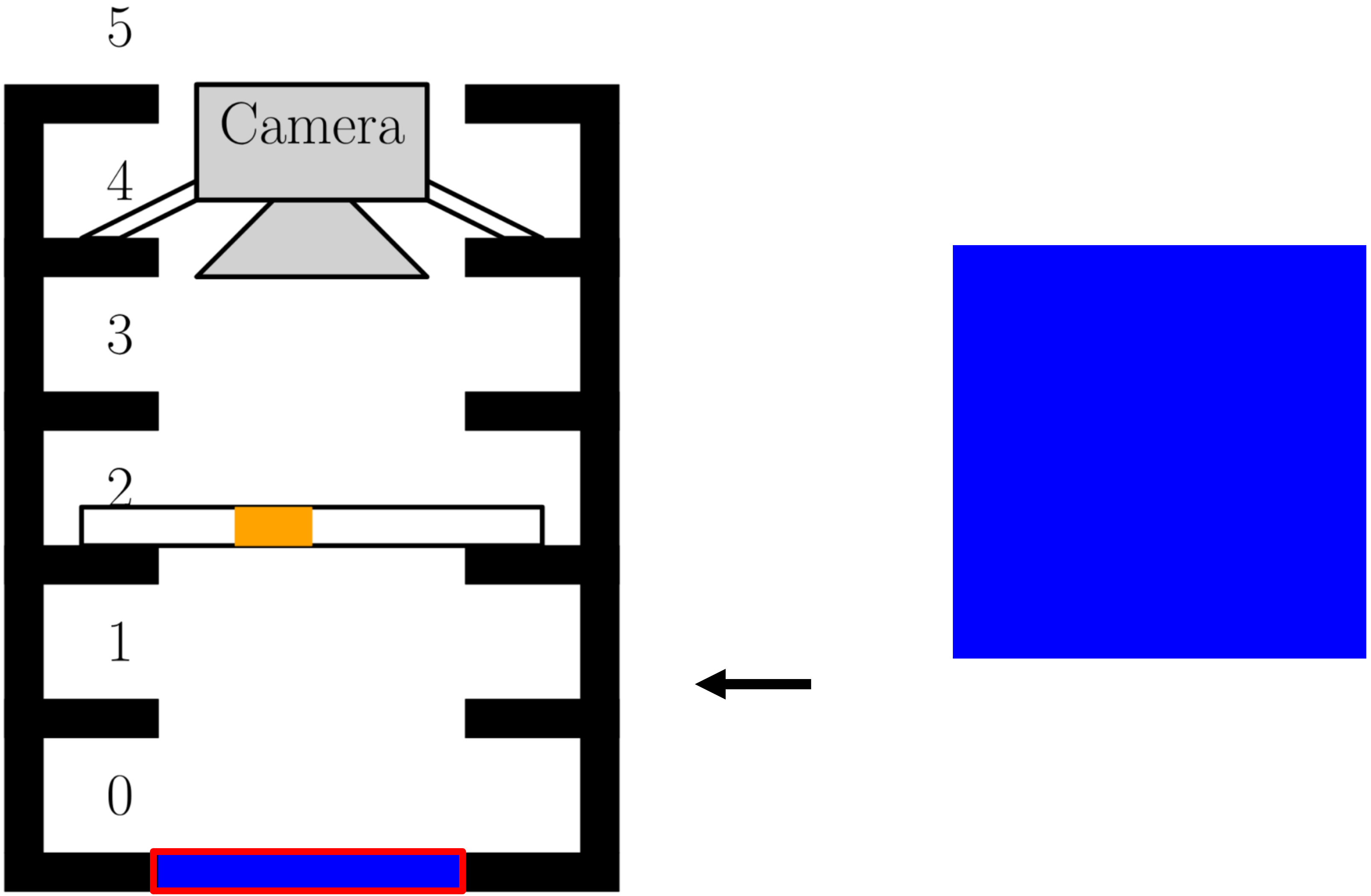
Implementierung

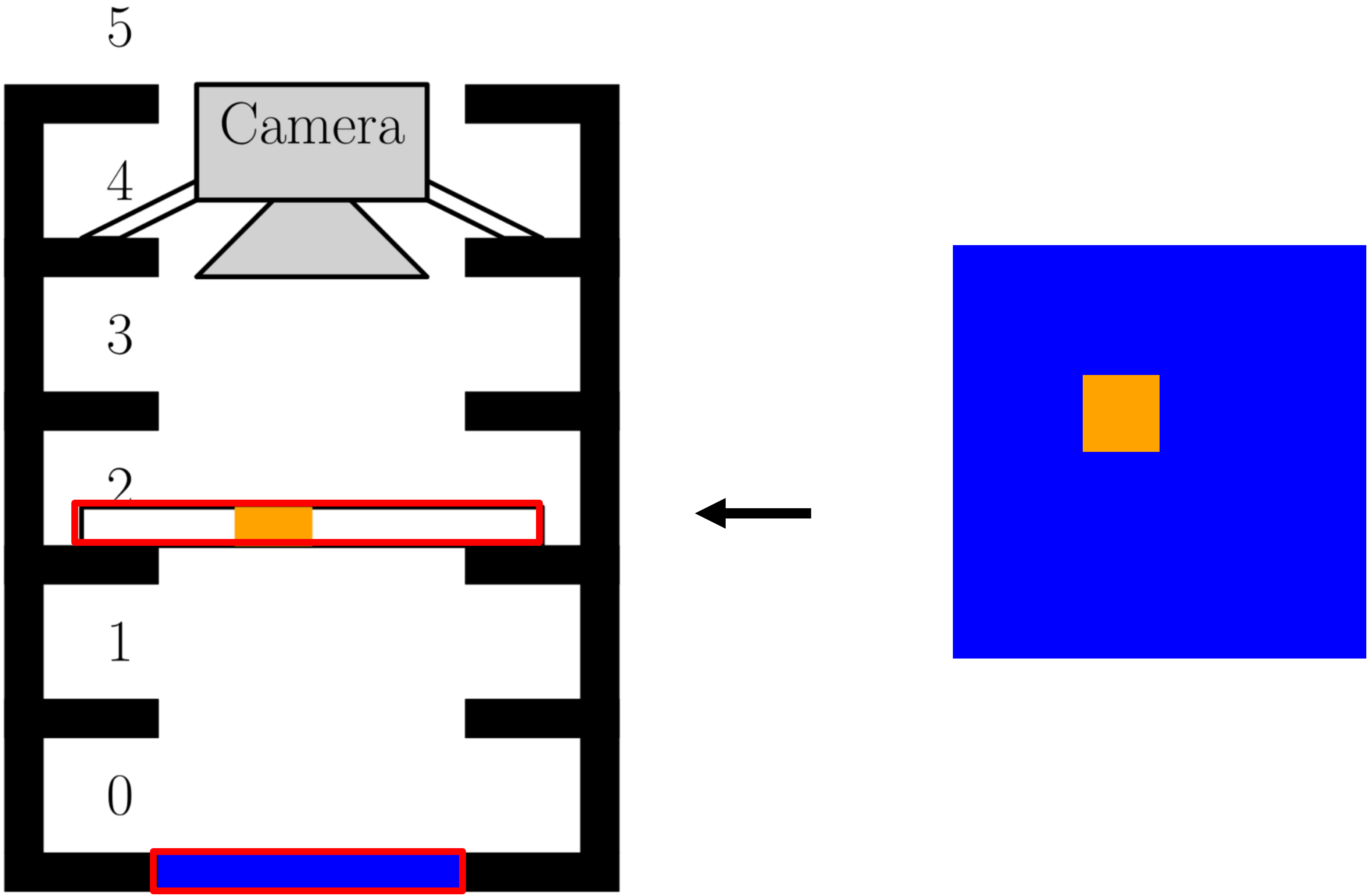
- `insertPlane(int height, int x0, int y0, int x1, int y1, int r, int g, int b)`: Diese Methode soll eine Glasplatte mit einem Rechteck oder einer Diagonale auf Ebene `height` in die Multiplan-Kamera einfügen. Der Rest der Glasplatte ist jeweils vollständig transparent. Falls es sich um eine Diagonale handelt, so ist $x1 = y1 = -1$. Sie dürfen davon ausgehen, dass die x- und y-Werte innerhalb der Matrix liegen (ausser im zuvor genannten Fall) und r,g und b Werte von 0 bis 255 haben. Die Rechtecke und Diagonalen funktionieren sonst analog zu Teilaufgabe a. Die Höhe `height` kann Werte von 0 bis 10 annehmen. Die neue Platte soll nur eingefügt werden, wenn sich nicht bereits eine Platte auf der spezifizierten Höhe befindet. Anderenfalls soll der Methodenaufruf einfach "ignoriert" werden - also keinen Effekt haben.

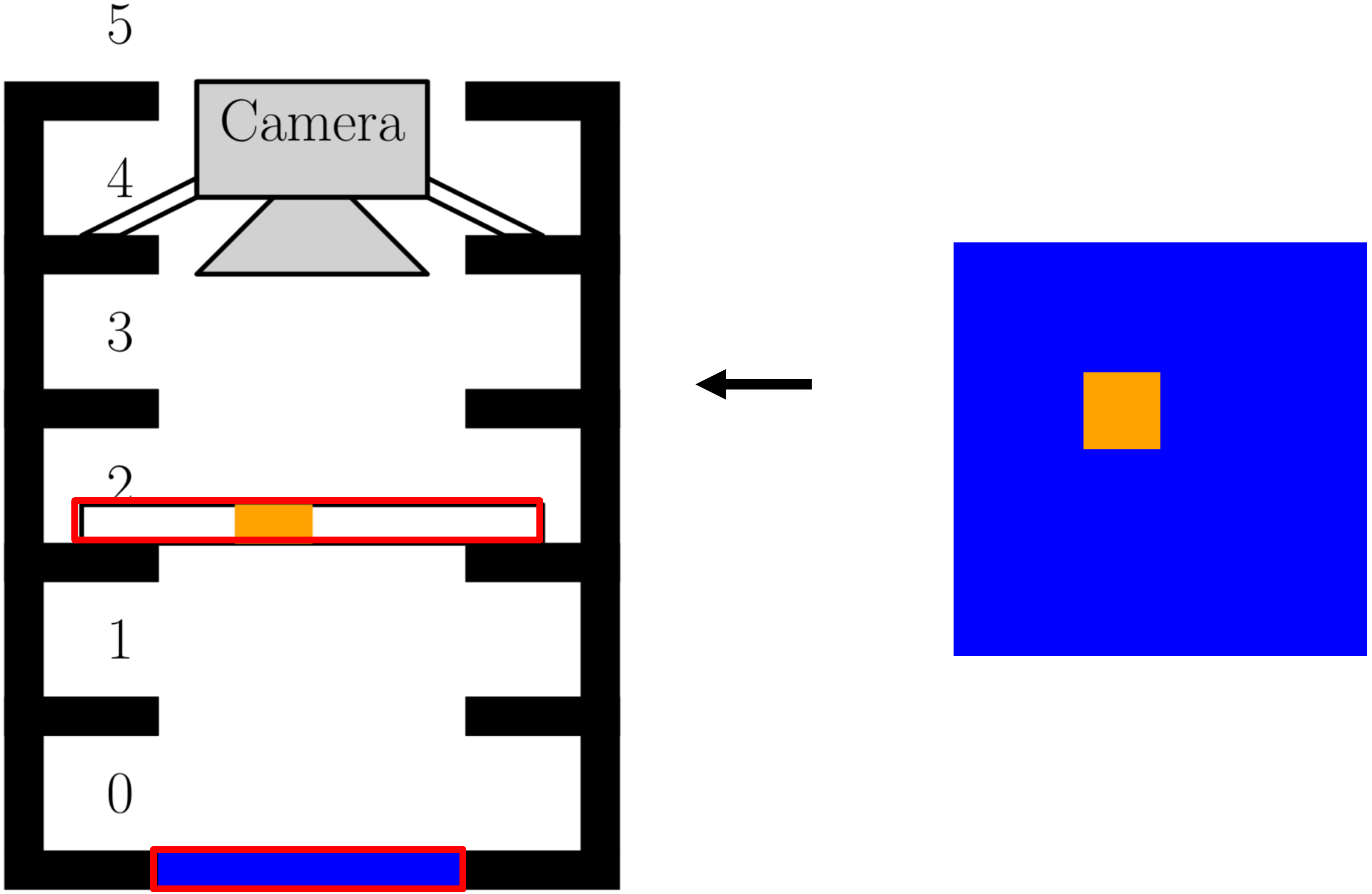
- `ImageFrame getPhoto(int height)`:
  Die Kamera wird temporär auf Höhe `height` platziert und soll eine Aufnahme nach unten machen. Rechtecke und Diagonalen auf den platzierten Ebenen können dabei die Figuren auf den darunterliegenden Ebenen und den Hintergrund (teilweise) überdecken - ein Pixel an Position (x,y) auf Höhe h überdeckt einen Pixel an derselben Position auf Höhe h′ falls $h' < h$. Ebenen ohne Glasplatten sind komplett transparent. Generieren Sie diese Bildaufnahme und geben Sie sie als ImageFrame zurück. Sollte sich auf Höhe `height` bereits eine Glasplatte befinden, geben Sie stattdessen `null` zurück. Sie dürfen annehmen, dass `height` nur Werte von 0 bis und mit 10 annimmt. Die Glasplatten und die Photos haben jeweils die Dimensionen des Hintergrundes.


5
Camera
4
3
2
1
0


5
Camera
4
3
2
1
0


5
Camera
4
3
2
1
0


5
Camera
4
3
2
1
0


5
Camera
4
3
2
1
0

Implementierung

Hoare Logic

$\{p\}\ s\ \{t\}$

```
{                }

if (x > y) {
  z = x - y
} else {
  z = y - x;
}

{z > 0}
```

Precondition {   }

```
if (x > y) {
  z = x - y
} else {
  z = y - x;
}
```

Postcondition {z > 0}

Stärkere und schwächere Aussagen

- Wir können *stärkere* und *schwächere* Aussagen unterscheiden
  - Wenn $P_1 \Longrightarrow P_2$ gilt, dann ist P_1 stärker als P_2 (und P_2 schwächer als P_1)

- Die stärkste Aussage ist `false`, da `false` alles impliziert
- Die schwächste Aussage ist `true`, **da `true` nur `true` impliziert**

Finde die Weakest Precondition:
Finde das schwächste p, sodass nach
Ausführung von `s` `t` gilt

Rückwärtsschliessen: Vorgehen

{Precondition}
Statement$_1$;
...
{Assertion$_{n-2}$}
Statement$_{n-1}$;
{Assertion$_{n-1}$}
Statement$_n$;
{Postcondition}

- **Start:** Wählen (wissen, raten) einer sinnvollen *Nachbedingung*
- **Schrittweise:** Herleiten der vorherigen Aussage (Assertion$_{i-1}$) durch Einbeziehen des *Effekts* der vorherigen Anweisung (Statement$_i$)
- **Ziel:** Herleiten einer *notwendigen und hinreichenden Vorbedingung*

- Rückwärts = «Welche Vorbedingung braucht mein Code, damit er die gewählten Garantien (Nachbedingung) geben kann?»

Weakest Precondition

- Die **schwächste Vorbedingung (weakest precondition)** ist die schwächste Vorbedingung, die die Postcondition impliziert.
  - Falls die Postcondition `{ true }` ist, so ist `{ true }` die schwächste Vorbedingung. Alles impliziert die Postcondition, also insbesondere auch die schwächste Bedingung `true`.
  - Falls die Postcondition `{ false }` ist, so ist `{ false }` die schwächste Vorbedingung. Nur `{ false }` impliziert die Postcondition, demensprechend ist es die schwächste (und einzige) Vorbedingung.
- Die vorgestellten Regeln fürs Rückwärtsschliessen ergeben direkt die schwächsten Vorbedingungen.

Weakest Precondition – Einfaches Beispiel

{ }

x = y*y

{x>4}

Weakest Precondition – Einfaches Beispiel

{ }

x = y*y ______ {x>4}

{x>4}

Weakest Precondition – Einfaches Beispiel

{ }

x = y*y ———— {y*y>4}

{x>4}

Weakest Precondition – Einfaches Beispiel

{y*y>4}

x = y*y ________

{x>4}

Weakest Precondition – Einfaches Beispiel

$\{|y|>2\}$

$x = y*y$ ________

$\{x>4\}$

Weakest Precondition – Zweites Beispiel

$\{ \qquad \}$

y = x+3

z = y+1

$\{z>4\}$

Weakest Precondition – Zweites Beispiel

{ }

$y = x+3$ ____________

$z = y+1$ ____________ $\{z>4\}$

$\{z>4\}$

Weakest Precondition – Zweites Beispiel

$\{ \qquad \}$

y = x+3 ________

z = y+1 ________ $\{y+1 > 4\}$

$\{z>4\}$

Weakest Precondition – Zweites Beispiel

{   }

y = x+3 ________ {y+1 >4}

z = y+1 ________

{z>4}

Weakest Precondition – Zweites Beispiel

{ }

$y = x+3$ ______ $\{x+3+1 > 4\}$

$z = y+1$ ______

$\{z>4\}$

Weakest Precondition – Zweites Beispiel

{ }

y = x+3 ________ {x > 0}

z = y+1 ________

{z>4}

Weakest Precondition – Zweites Beispiel

$\{x > 0\}$

$y = x+3$ __________

$z = y+1$ __________

$\{z>4\}$

Weakest Precondition – Aufgabe 1

```
{                 }

   x = a - 4;

{x>0}
```

Weakest Precondition – Aufgabe 2

```
{              }

   x = y + z;

   y = y - 5;

{y < 0}
```

Weakest Precondition – Prüfungsbeispiel

```
{                }

  w = a

  x = w + b;

  y = x * 2;

{y > 0 && b > 10}
```

Weakest Precondition – Prüfungsbeispiel

```
{                    }

 p = 3 * q

 p = p + 1;

{p > 15}
```

If-Anweisungen und Aussagen

- **Ziel:** Regel für Tripel `{P} if (b)` S_1 `else` S_2 `{Q}`

- **Beobachtungen**
  - Ausführung von S_1 wenn b hält
  - Ausführung von S_2 wenn $\neg b$ hält
  - P hält in beiden Fällen vor der Ausführung
  - Q muss in beiden Fällen nachher gelten

```
{P}
  if (b) {
    S1;
  } else {
    S2;
  }
{Q}
```

Hoare-Logik-Regel für if-Anweisungen

- Das Tripel `{P} if (b)` S_1 `else` S_2 `{Q}` ist gültig, genau dann wenn
  1. $\{P \land b\}\ S_1\ \{Q\}$ gültig ist
  2. $\{P \land \neg b\}\ S_2\ \{Q\}$ gültig ist

Vorgehen für if-Anweisungen

- **Situation:** Entscheide, ob `{P} if (b)` S_1 `else` S_2 `{Q}` gültig ist

- **Empfohlenes Vorgehen:**
  1. Wende bekannte Regeln wie rechts gezeigt an (auf S_1 und S_2)
  2. Zeige notwendige Implikationen
     1. $P \land b \implies R_1$
     2. $P \land \neg b \implies R_2$

```
if (b) {
  {P ∧ b}
  {R1}
    S1;
  {Q}
} else {
  {P ∧ ¬b}
  {R2}
    S2;
  {Q}
}
```

If-Anweisungen im Kontext: Beispiel

Zu zeigen:
Gültigkeit des Tripels

```
{n > 2}
  m = n + 3;
  if (m % 2 == 0) {
    m = m / 2;
  } else {
    m = m + 1;
  }
  k = m + n;
{k > 5}
```

Rückwärts vorgehen:

```
   {n > 2}
     m = n + 3;
   {P}
     if (m % 2 == 0) {
2.     {P ∧ m % 2 ==0}
       ⇒{(m / 2) + n > 5}
3.       m = m / 2;
2.     {m + n > 5}
     } else {
       {P ∧ ¬(m % 2 ==0)}
       ⇒{(m + 1) + n > 5}
3.       m = m + 1;
2.     {m + n > 5}
     }
   {m + n > 5}
1.   k = m + n;
   {k > 5}
```

Offene Frage:
Wie bestimmen wir ***P*** (die Vorbedingung der if-Anweisung)?

Systematisch rückwärts raten

```
{P}
if (b) {
   {P ∧ b}
   {R₁}
      S₁;
   …
} else {
   {P ∧ ¬b}
   {R₂}
      S₂;
   …
}
…
```

1. **Beobachtung:**
   a. S_1 wird nur ausgeführt, falls b hält
   b. S_2, falls ¬b hält

2. **Konsequenz:**
   a. R_1 muss daher nur halten, wenn b hält
   b. R_2, falls ¬b hält

3. **Systematische Wahl für *P* daher:**

$$(b \implies R_1) \land (\neg b \implies R_2)$$

Systematische Wahl richtig?

```
{P}
if (b) {
   {P ∧ b}
⇒  {R1}
     S1;
   …
} else {
   {P ∧ ¬b}
⇒  {R2}
     S2;
   …
}
…
```

- Systematische Wahl für ***P***:

  $$(b \Longrightarrow R_1) \ \wedge \ (\neg b \Longrightarrow R_2)$$

- **Konsequenz:** Es halten die notwendigen Implikationen

  1. $P \wedge \mathrm{b} \quad \Longrightarrow \quad R_1$,
     also $((b \Longrightarrow R_1) \ \wedge \ (\neg b \Longrightarrow R_2)) \wedge \mathrm{b} \ \Longrightarrow \ R_1$

  2. Analog für $P \wedge \neg \mathrm{b} \ \Longrightarrow \ R_2$

Beispiel Schritt für Schritt

v1

```
{n > 2}

m = n + 3;

if (m%2==0) {
  m = m / 2;
} else {
  m=m+1;
}
{m + n > 5}
k = m + n;
{k > 5}
```

v2

```
{n > 2}

m = n + 3;

if (m%2==0) {
  m = m / 2;
  {m + n > 5}
} else {
  m=m+1;
  {m + n > 5}
}
{m + n > 5}
k = m + n;
{k > 5}
```

v3

```
{n > 2}

m = n + 3;

if (m%2==0) {
  {(m/2) + n > 5}
  m = m / 2;
  {m + n > 5}
} else {
  {(m+1) + n > 5}
  m=m+1;
  {m + n > 5}
}
{m + n > 5}
k = m + n;
{k > 5}
```

v4

```
{n > 2}

  m = n + 3;
{(m%2=0 ⟹ (m/2) + n > 5)
∧(m%2=1 ⟹ (m+1) + n > 5)}
  if (m%2==0) {
    {(m/2) + n > 5}
    m = m / 2;
    {m + n > 5}
  } else {
    {(m+1) + n > 5}
    m=m+1;
    {m + n > 5}
  }
  {m + n > 5}
  k = m + n;
{k > 5}
```

v4 ➡

```
{n > 2}
{((n+3)%2=0 ⟹ ((n+3)/2) + n>5)
∧((n+3)%2=1 ⟹ ((n+3)+1) + n>5)}
  m = n + 3;
{(m%2=0 ⟹ (m/2) + n > 5)
∧(m%2=1 ⟹ (m+1) + n > 5)}
  if (m%2==0) {
    {(m/2) + n > 5}
    m = m / 2;
    {m + n > 5}
  } else {
    {(m+1) + n > 5}
    m=m+1;
    {m + n > 5}
  }
  {m + n > 5}
  k = m + n;
{k > 5}
```

v5

Dann noch zu zeigen:

$$(n > 2)$$

$$\Longrightarrow$$

$$\begin{aligned}(\quad & ((n+3)\%2=0 \Longrightarrow ((n+3)/2) + n > 5) \\ \wedge \quad & ((n+3)\%2=1 \Longrightarrow ((n+3)+1) + n > 5))\end{aligned}$$

Fallunterscheidung (zwei Konjunkte):

1. Wenn $n > 2$ und $(n+3)\%2=0$, dann $(3n+3)/2 > 5$ ✓
2. Wenn $n > 2$ und $(n+3)\%2=1$, dann $2n + 4 > 5$ ✓

Weakest Precondition mit Verzweigung

Schwächste Vorbedingung - Beispiel

```
{               }

  if (x >= y){
    y = x
  }

{y >= x}
```

Weakest Precondition – Mit Verzweigung

```
{              }
if (x<5) {
        y = x*x;
} else {
        y = x+1;
}

{y>9}
```

Weakest Precondition – Mit Verzweigung

{ }

```
if (x<5)
        y = x*x;
else
        y = x+1;
```

{y>9}

{x<5}

{x>=5}

Weakest Precondition – Mit Verzweigung

{ }

```
if (x<5)
        y = x*x;
else
        y = x+1;
```

$\{x<5\}$

$\{y>9\}$

$\{x>=5\}$

$\{y>9\}$

Weakest Precondition – Mit Verzweigung

{ }

```
if (x<5)
        y = x*x;
else
        y = x+1;
```

$\{x<5\}$

$\{y>9\}$

$\{x>=5\}$

$\{x+1>9\}$

Weakest Precondition – Mit Verzweigung

{ }

```
if (x<5)
        y = x*x;
else
        y = x+1;
```

{x<5}

{y>9}

{x>=5}

{x+1>9}

Weakest Precondition – Mit Verzweigung

{ }

```
if (x<5)
        y = x*x;
else
        y = x+1;
```

{x<5}

$\{x*x>9\}$

{x>=5}

$\{x+1>9\}$

$$\{(\mathbf{x<5} \;\&\&\; x*x>9) \;||\; (\mathbf{x>=5} \;\&\&\; x+1>9)\}$$

{ }

```
if (x<5)
        y = x*x;
else
        y = x+1;
```

{x<5}

$\{x*x>9\}$

{x>=5}

$\{x+1>9\}$

Weakest Precondition - Prüfungsbeispiel

$\{ \qquad \qquad \}$

```
if (x > y)
    z = x – y;
else
    z = y – x;
```

$\{ \; z > 0 \; \}$

Weakest Precondition - Prüfungsbeispiel

{ }

```
if (x > y)
        z = x – y;
else
        z = y – x;
```

{ z > 0 }

{x > y}

{x <= y}

Weakest Precondition - Prüfungsbeispiel

{ }

```
if (x > y)
        z = x – y;
else
        z = y – x;
```

{ z > 0 }

{x > y}

{z > 0}

{x <= y}

{z > 0}

Weakest Precondition - Prüfungsbeispiel

{ }

```
if (x > y)
        z = x – y;
else
        z = y – x;
```

{ z > 0 }

{x > y}

{z > 0}

{x <= y}

{z > 0}

Weakest Precondition - Prüfungsbeispiel

$\{ \qquad \qquad \}$

```
if (x > y)
        z = x – y;
else
        z = y – x;
```

$\{ \ z > 0 \ \}$

{x > y}

$\{x - y > 0\}$

{x <= y}

$\{y - x > 0\}$

Weakest Precondition - Prüfungsbeispiel

$\{ \qquad \qquad \}$

```
if (x > y)
        z = x – y;
else
        z = y – x;
```

$\{\ z > 0\ \}$

$\{x > y\}$	**$\{x <= y\}$**
$\{x - y > 0\}$	$\{y - x > 0\}$

Weakest Precondition - Prüfungsbeispiel

$\{\qquad\qquad\}$

```
if (x > y)
        z = x – y;
else
        z = y – x;
```

$\{\ z > 0\ \}$

{x > y}

$\{x > y\}$

{x <= y}

$\{y - x > 0\}$

$\{((\mathbf{x > y})\ \&\&\ (x > y))\ ||\ ((\mathbf{x <= y})\ \&\&\ (y > x))\}$

$\{\qquad\qquad\}$

if (x > y)
    z = x – y;
else
    z = y – x;

$\{\ z > 0\ \}$

$\{\mathbf{x > y}\}$

$\{x > y\}$

$\{\mathbf{x <= y}\}$

$\{y > x\}$

Gültig/Ungültig

Hoare Tripel – Prüfungsbeispiel HS18

```
{ b > c }
 if (x > b) {
     a = x;
} else {
     a = b;
}
{ a > c }
```

Hoare Tripel – Prüfungsbeispiel HS18

```
{ true }
 if (x > y) {
     y = x;
} else {
     y = -x;
}
{ y >= x}
```

Hoare Tripel – Prüfungsbeispiel HS18

```
{ x > 0 }
    y = x * x;
    z = y / 2;
{ z > 0 }
```

Prüfungsbeispiel HS21

1. WP: { }

```
k = m * 3;
Q: { k > 0 }
```

Prüfungsbeispiel HS21

```
2. WP: {                                                  }


   x = y * 2;
   x = x + 1;
   Q: {x > 2 }
```

Prüfungsbeispiel HS21

```
3. WP: {                                                    }

   if (a > b) {
     c = (-2) * a;
   } else {
     c = a + 4;
   }
   Q: { c > 0 }
```

Übung 7 – Relevanteste Aufgaben (A.o.G)

- Aufgabe 1 (!!)
- Aufgabe 2 (!!!)
- Aufgabe 3 (!!!)
- Aufgabe 4 (!)
- Bonus (!!!)