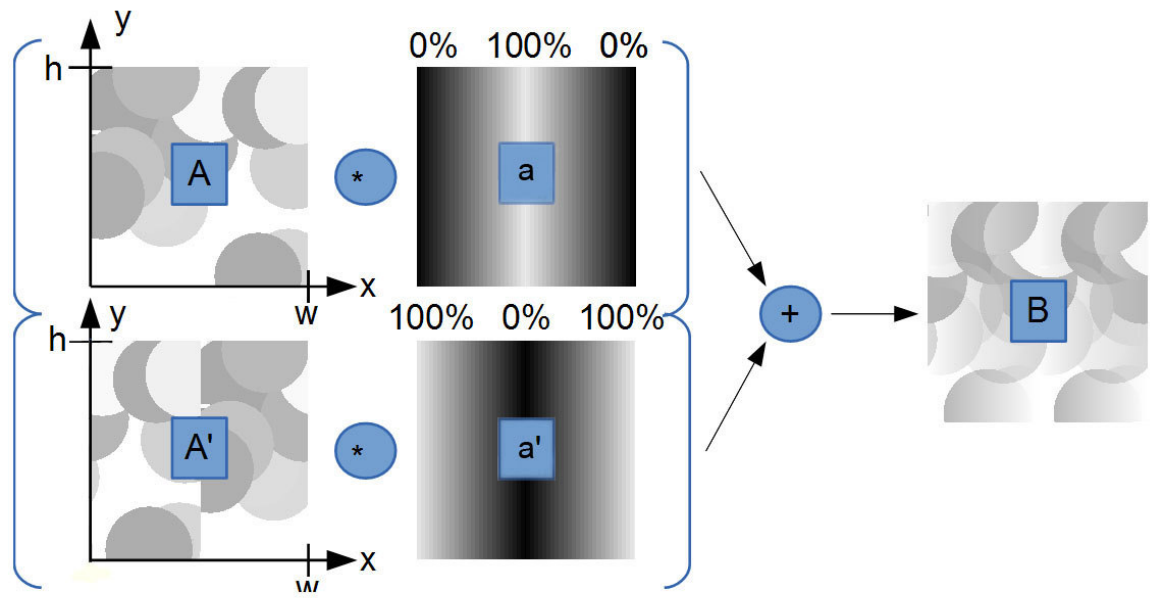
Prüfungsaufgabe 1

Bei der Texturierung gibt es häufig Situationen, in denen Texturen mehrfach wiederholt aneinander gereiht werden müssen (z.B. bei Rasen, Schotterplätzen,…). Dabei ist es wichtig, dass die Texturen so beschaffen sind, dass die Nahtstellen nicht sichtbar sind.

Bei Texturen, die mit Kameras aufgenommen wurden, ist dies in der Regel zunächst nicht gegeben. Sie müssen deshalb entsprechend aufbereitet werden. Das kann z.B. mit dem im Folgenden beschriebenen Verfahren erreicht werden.

Das Ausgangsbild **A** wird zunächst zyklisch um die halbe Bildbreite **w** horizontal verschoben. Das Ergebnis ist Bild **A'**. Die beiden Bilder **A** und **A'** werden nun so überblendet, dass am linken und rechten Rand zu 100% **A'** und in der Mitte zu 100% **A** verwendet wird. Wie in der Skizze 1 illustriert, entsteht das Ergebnisbild B durch:

$$B = (A * a) + (A' * a') \quad \text{mit} \quad a'(x,y) = \left| \frac{-2x}{w-1} + 1 \right| \quad \text{und} \quad a(x,y) = 1 - a'(x,y)$$



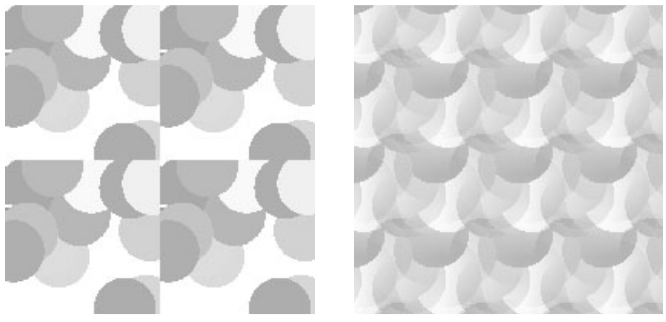
Skizze 1

Das Ergebnisbild B lässt sich nun horizontal beliebig aneinanderreihen.

Nun verfährt man ausgehend von Bild **B** in vertikaler Richtung analog, indem man das Bild in y-Richtung um die halbe Höhe zyklisch verschiebt und mit dem Originalbild **B** überblendet. Die Blendmasken **a** und **a'** sind dabei um 90° gedreht.

Skizze 2:

Links: 2x2-Wiederholung des Originalbildes mit deutlich sichtbaren Nahtstellen
Rechts: 2x2-Wiederholung des überblendeten Bildes



Prüfungsaufgabe 1

Aufgabe:

Implementieren Sie die Berechnung wiederholbarer Texturen in OpenCL nach dem o.g. Verfahren. Gehen Sie dabei wie folgt vor:

- Integrieren Sie das Verfahren in das Bildverarbeitungs-Rahmenprogramm *ImageFX.*
- Arbeiten Sie mit Arrays, nicht mit CL-Image-Sampler
- Verwenden Sie jeweils ein Workitem für jedes Pixel. Wählen Sie eine geeignete Workgroup-Größe.
- Die Blendmasken **a** und **a'** sollen nicht als Bilder vorgegeben oder erzeugt werden. Die benötigten Werte sollen in den Kerneln immer dann direkt berechnet werden, wenn sie benötigt werden. Dasselbe gilt für die zyklisch verschobenen Bilder **A'**.
- Implementieren Sie zwei separate Kernel für die horizontale und die vertikale Überblendung, z.B.:

  ```
  kernel void xblend(global uchar4 *in, global uchar4 *out)
  kernel void yblend(global uchar4 *in, global uchar4 *out)
  ```

  Da eine `barrier` nur auf die Workitems innerhalb einer Workgroup aber nicht über Workgroup-Grenzen hinweg wirkt, ist es nicht möglich, innerhalb eines Kernels zunächst horizontal zu blenden, dann eine `barrier` zu setzen und anschließend vertikal zu blenden. Eine solche Synchronisation muss von der CPU aus gesteuert werden, indem 2 Kernel separat aufgerufen werden.

Bonusaufgabe:

Realisieren Sie die Überblendung in horizontaler und vertikaler Richtung gemeinsam in einem einzigen Kernel (Kombination von `xblend` und `yblend`). Wie oben bereits beschrieben, ist es dabei *nicht* möglich, in einem Kernel zwischen den beiden Blendvorgängen eine Synchronisation vorzunehmen. Das Ergebnis des ersten Blendvorgangs wird deshalb nicht als Zwischenergebnis gespeichert. Stattdessen wird beim zweiten Blendvorgang der Zugriff auf ein Pixel des Zwischenergebnisses durch dessen (on-the-fly-) Berechnung ersetzt. Bei dieser Vorgehensweise werden die Berechnungen des ersten Blendvorgangs doppelt ausgeführt (weil beim zweiten Blendvorgang auf jedes Pixel des Zwischenergebnisses zwei Mal zugegriffen wird), man spart sich dadurch aber einen Kernel-Aufruf. Bei vielen Anwendungen führt eine solche Strategie zu einer Beschleunigung, weil die GPUs sehr viel Rechenpower besitzen und Kernelaufrufe mit Latenzen verbunden sind.

Hinweise:

- Benutzen Sie den %-Operator um z.B. bei der zyklischen Verschiebung Koordinaten in den Bildbereich zu mappen.
- OpenCl-Funktionen, die für diese Aufgabe hilfreich sein können sind: `convert_float4(…), clamp(…), convert_uchar4(…), mix(…)`
- Testen Sie Ihr Ergebnis, indem Sie im letzten Kernel das Ergebnis mit einer 2*2-Wiederholung ausgeben. Dazu benötigen Sie eine Koordinatentransformation in der Art: `int X=(GX*2)%GW, Y=(GY*2)%GH;` Unerwünschte Nahtstellen werden bei der Ausgabe dann sichtbar.