Aufgabe 14

Es war ein generisches Subdivisionsverfahren zu implementieren, dass mit beliebigen Gewichtsvektoren als Eingabe umgehen kann und die gegebenen Polygonzüge entsprechend verfeinert.
Die graphische Oberfäche war ja schon vorgegeben, so dass wir nur die Methode `private void subdivide(PdVector[] old_points, PdVector[] new_points, double[] mask)` der `MySubdivisionProject`-Klasse umsetzen mussten.

Die Implementierung sieht dabei folgendermaßen aus:

```
private void subdivide(PdVector[] old_points, PdVector[]
    new_points, double[] mask){
  PdVector[] temp_points = new PdVector[new_points.length];

  // subdivision step
  for(int i=0; i<old_points.length; i++){
    new_points[2*i] = PdVector.copyNew(old_points[i]);
    new_points[2*i+1] = PdVector.blendNew(0.5, old_points[i],
        0.5,
        old_points[(i+1)%old_points.length]);
  }

  // averaging step, using the mask
  for(int i=0; i<new_points.length; i++){
    PdVector vec_sum = new PdVector(new_points[0].getSize());

    for (int m = 0; m < getMaskLength(); m++) {
      if(mask[m] == 0) continue;
      // determine position, make sure to stay in the array
      int pos = (i - m_mask_center + m) % new_points.length;
      if(pos < 0) pos = new_points.length + pos;

      PdVector vec = PdVector.copyNew(new_points[pos]);
      // scale with corresponding weight
      vec.multScalar(mask[m]);
      // add up
      vec_sum.add(vec);
    }
    // save temporarily
    temp_points[i] = vec_sum;
    temp_points[i].multScalar(1.0/m_mask_normalization.
        getValue());
  }
  // copy temporary points back
  for(int i=0; i<new_points.length; i++)
    new_points[i] = temp_points[i];

}
```

Zunächst wird die Anzahl der Punkte verdoppelt, in dem zusätzlich zu den alten Punkten jeweils die Mittelpunkte zwischen zwei benachbarten Punkten eingefügt werden. Diese neuen Punkte werden dann entsprechend der Maske gewichtet. Für jeden neuen Punkt wird die Maske zentriert über diesen Punkt gelegt (an den Rändern muss aufgepasst werden, dass man nicht die Menge verlässt) und alle Punkte unter der Maske entsprechend der Gewichte aufsummiert. Damit wir bei diesem Mittelungsschritt nicht versehentlich schon gemittelte neue Punkte verwenden, geschieht dies auf einem temporären Array.

Sind alle Punkte so bearbeitet, kann das Ergebnis, nachdem noch normiert wurde, richtig gespeichert werden.

Die Implementierung ist nun noch mit dem Chaikin und dem DLG-Verfahren zu testen:

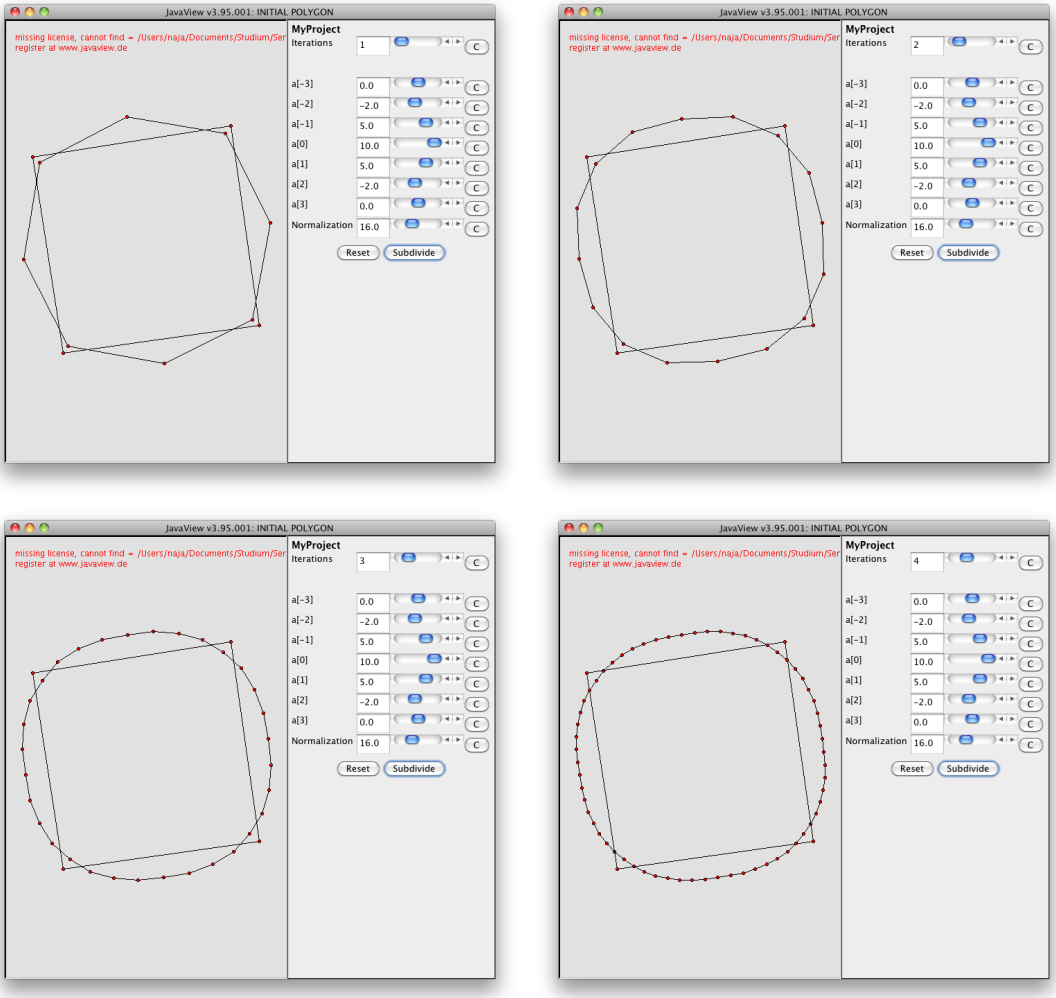

Abbildung 1: DLG-Verfahren, 1 bis 4 Schritte

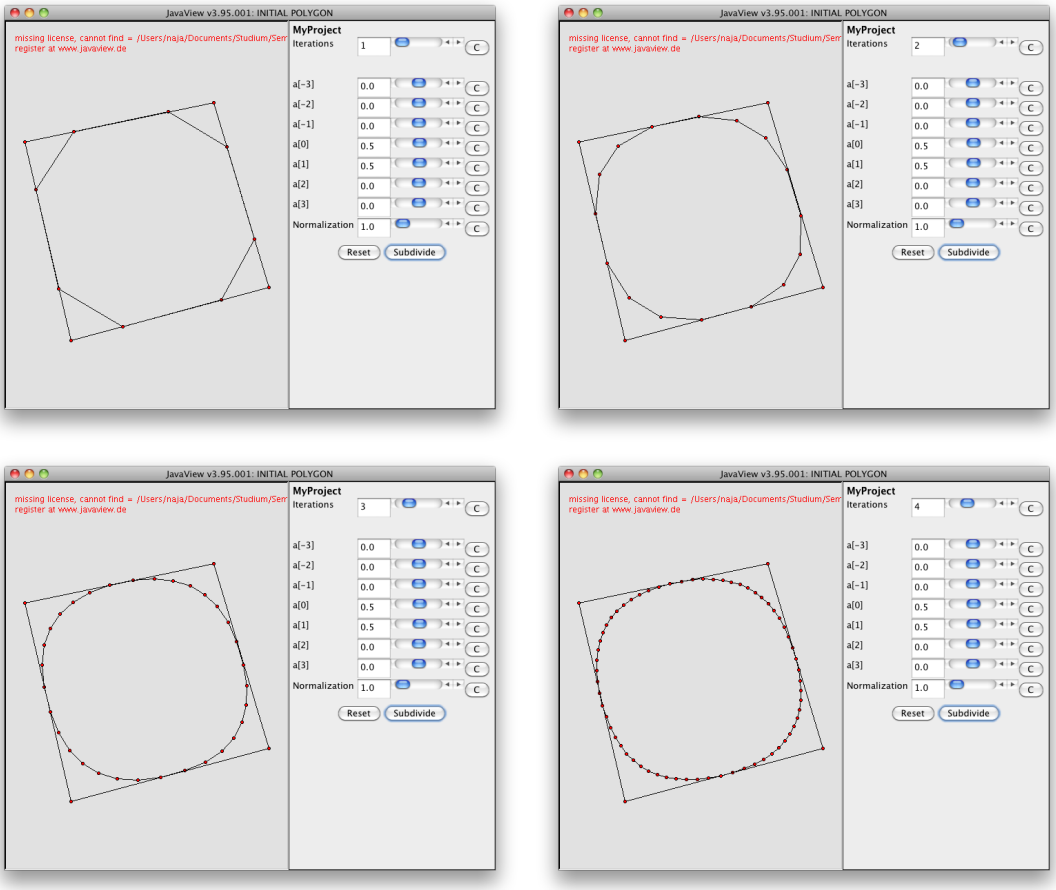

Abbildung 2: Chaikin-Verfahren, 1 bis 4 Schritte