Aufgabe 1: 2D-Vektorgrafiken/Projektionen (7 Punkte)

(a) Erklären Sie, wie die Begriffe Computergrafik und Modellierung sowie Bilderkennung und Bildbearbeitung zusammenhängen. Verdeutlichen Sie Ihre Erklärungen durch ein Schaubild. **(3 Punkte)**

(b) Welche Sequenz beschreibt die folgende Transformation vom Ausgangszustand (links) in den Endzustand (rechts)? **(2 Punkte)**

Hinweis: $T(t_x, t_y)$ bezeichnet eine Verschiebung um den Vektor (t_x, t_y), $S(s_x, s_y)$ eine Skalierung um die Faktoren (s_x, s_y), und $R(\alpha)$ eine positive Rotation um den Winkel α.

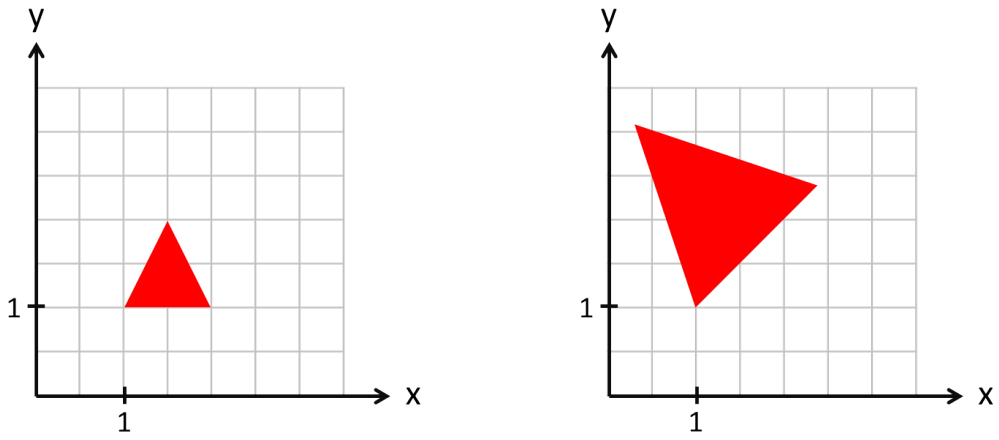


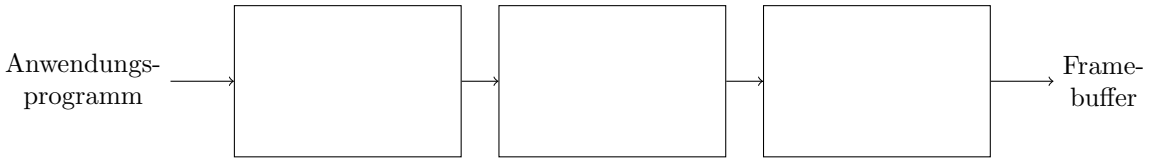
Abbildung 1: Dreieck vor der Transformation (links) und nach der Transformation (rechts)

- ☐ $T(-1,-1) \cdot S(2,2) \cdot R(45) \cdot T(1,1)$
- ☐ $T(-1,-1) \cdot S(2,2) \cdot R(-45) \cdot T(1,1)$
- ☐ $T(1,1) \cdot R(45) \cdot S(2,2) \cdot T(-1,-1)$
- ☐ $S(2,2) \cdot T(1,1) \cdot R(45) \cdot T(-1,-1)$

(c) Für einen speziellen Bildschirm sei das Screen-Koordinatensystem zwischen (0,0) und (512,512) definiert. Wie berechnen sich für ein Fragment mit den Screen-Koordinaten (x_s, y_s) die zugehörigen normalisierten Gerätekoordinaten (x_n, y_n)? **(2 Punkte)**

Aufgabe 2: WebGL (7 Punkte)

(a) Beschriften Sie folgendes Schaubild der WebGL-Rendering-Pipeline mit den richtigen Begriffen und beschreiben Sie anhand der drei Stufen kurz den Ablauf eines WebGL-Programms. **(3 Punkte)**



(b) Welche Stufe der Rendering-Pipeline ist in WebGL 2.0 nicht frei programmierbar? **(1 Punkt)**

- ☐ Beleuchtungsberechnung
- ☐ Vertextransformation
- ☐ Rasterisierung
- ☐ Projektion

(c) Welche Funktionen werden im folgenden Vertex-Shader umgesetzt? Kreuzen Sie **alle** korrekten Antworten an. (Gehen Sie davon aus, dass die Namen der Eingabevariablen passend zu ihrem Inhalt gewählt wurden.) **(3 Punkte)**

```
<script id="vertex-shader" type="x-shader/x-vertex">#version 300 es
   in vec4 vPosition;
   in vec4 vNormal;

   uniform mat4 modelMatrix;
   uniform mat4 viewMatrix;
   uniform mat4 projectionMatrix;

   void main()
   {
      mat4 var1 = viewMatrix * modelMatrix;
      mat4 var2 = inverse(transpose(var1));
      vec4 var3 = var2 * vNormal;
      gl_Position = projectionMatrix * var1 * vPosition;
   }
</script>
```

- ☐ Transformation der Vertexposition von Objekt- in Screenkoordinaten.
- ☐ Transformation der Vertexnormalen von Objekt- in Kamerakoordinaten.
- ☐ Transformation der Vertexposition von Objekt- in Clipkoordinaten.
- ☐ Transformation der Vertexnormalen von Objekt- in Normalisierte Gerätekoordinaten.
- ☐ Speichern der finalen Vertexposition in einer Built-In Variable.
- ☐ Übergabe der finalen Vertexnormalen vom Vertex- zum Fragment-Shader.