מימוש פוקוס בתמונה

על מנת שנשיג עומק שדה, ממשנו את פעולת הצמצום וההרחבה של האישונים על המצלמה .הגדרנו משטח חדש בשם משטח מיקוד (focal plane) שמוצב מול משטח הצפייה. הנקודה על משטח המיקוד המתקבלת על ידי הקרן שנשלחת מהנקודה על משטח הצפייה נקראת נקודת מיקוד .

הגדרנו את צמצם המצלמה. מדובר בריבוע על משטח הצפייה שמתאר את "כמות" החלון שדרכו עובר האור. עבור כל נקודה על קצוות הצמצם שלחנו קרן שתעבור דרך נקודת המיקוד .אם כל הקרניים ששלחנו מהצמצם דרך נקודת המיקוד מחזירות אותו צבע, סימן שהאובייקט נמצא בפוקוס ואת הצבע שחזר נשים בפיקסל. אחרת, נערבב את הצבעים שחזרו (מה שיוצר את האשליה שהתמונה לא בפוקוס) ואת צבע זה נשים בפיקסל.

קטע קוד רלוונטי:

הוספנו למחלקה camera 2 מאפיינים חדשים:

1. Aperture שמגדיר את גודל הריבוע על משטח הצפיה
2. focalLenght שמגדיר את מרחק המשטח מיקוד מהמשטח צפייה

ועשינו refactor לפונקציה constructRaysThroughPixel שתחזיר מערך של rays

```
/*********** Administration ***********/
/**
 *
 * The function calculate one ray for ,but if there is focus calculates a list of ray given the parameters of the screen
 *
 * @param Nx
 *          number of pixels in the width
 * @param Ny
 *          number of pixels in the height
 * @param i,j
 *          index of the viewPlane where the ray
 * @param screenDistance
 *          distance from the camera to the viewplane
 * @param width
 *          width of the viewplane
 * @param height
 *          height of the viewplane
 * @return list of ray from the the viewplane to focus point if there is focus else from the camera to the viewplane
 */

public List<Ray> constructRaysThroughPixel(int Nx, int Ny, int i, int j, double screenDistance,
double width,
                                           double height) {
    List<Ray> rayList = new ArrayList<Ray>();
    Point3D Pij = p0.addVec(vTo.scalarMult(screenDistance));
    double Ry = height / Ny;
    double Rx = width / Nx;
    double dx = (i - (Nx + 1) / 2) * Rx;
    if (!Coordinate.ZERO.equals(dx))
        Pij = Pij.addVec(vRight.scalarMult(dx));
    double dy = (j - (Ny + 1) / 2) * Ry;
    if (!Coordinate.ZERO.equals(dx))
        Pij = Pij.addVec(vUp.scalarMult(-dy));
    Vector Vij = (Pij.subVec(p0)).normalize();
    Ray cameraRay = new Ray(p0, Vij);
    if (focalLength <= 0)
        rayList.add(cameraRay);
    else {
        double distanceToPij = Pij.distance(p0);
        Point3D fp = intersectionFocalPlane(cameraRay.getV(), screenDistance, distanceToPij);//
point on the focus plane
        double halfAperture = aperture / 2;
        for (int k = 0; k < NUM_RAYS_FOCUS; k++) {
            double randValue1 = -halfAperture + (2 * halfAperture) * r.nextDouble();
            double randValue2 = -halfAperture + (2 * halfAperture) * r.nextDouble();
            Vector v1Right = getVRight().scalarMult(randValue1);
            Vector v2Up = getVUp().scalarMult(randValue2);
            Point3D headRay = Pij.addVec(v1Right).addVec(v2Up);
            rayList.add(new Ray(headRay, fp.subVec(headRay)));
        }
    }
    return rayList;
}
```

```java
/**
 * Calculates the intersection with the focal plane
 * This is the focus point
 *
 * @param direction from the camera to the plane
 * @param screenDistance distance to the view plane
 * @param distancePixel distance to the pixel
 * @return the point of the intersection
 */
public Point3D intersectionFocalPlane(Vector direction, double screenDistance, double distancePixel) {
    Vector v = direction.normalize();
    double scalar = distancePixel * (screenDistance + focalLength) / screenDistance;
    v = v.scalarMult(scalar);
    return p0.addVec(v);
}
```