Quarcles: Definition and Simulation

F. T. Suttmeier

Department Mathematik
Universität Siegen
D-57258 Siegen, Germany

This article is distributed under the Creative Commons by-nc-nd Attribution License. Copyright © 2020 Hikari Ltd.

Abstract

In this note we present a definition for quarcles, roughly spoken intersections and overlays of quads and circles. Furthermore we present a related software tool as a part of our research in teaching programming.

Keywords: Teaching programming, Lean Software-Engineering, Structured Programming

1 Introduction

In this note we present a lean solution to a statement of task, intended to be given to study groups in highschool and undergraduate courses (c.f. Börstler & Sperber [1]):

A geometric figure in a two-dimensional plane
is a triangle,
a quad
or an overlay
or an intersection of two geometric figures.
Write a Program Geometric Figures!
Design a system with the possibility to create geometric figures and
to check if a given point is inside the area of a chosen geometric figure.

It turned out, as seen so often, that OOP is not a panacea. Even nearly graduate students struggle in designing simple software systems. This coincides with the author’s observation over the last decade that more and more
simple projects in study groups of undergraduate courses fail. Often there are already severe problems on designing simple user input and output. Partly this is caused by state of the art languages together with large libraries for colourful, clickable user-interaction. These seductive possibilities are very helpful for students to avoid solving the original task. The sense for pulling out the essentials from irrelevant details is weakly shaped.

*The ability to abstract decreases increasingly.*

Teaching programming can be well done by presenting appropriately chosen examples, Wirth [7]. In this sense, we are collecting material for our study groups, and the work at hand presents a further item in our pool of examples teaching the power of simplicity.

## 2 Quarcles

While studying literature on education in computer science, for the first time the author met the concept described in the introduction, which in this note is called *quarcles*. Roughly spoken, these objects are obtained by intersections and overlays of *quads* and *circles*. Presentation at hand proposes a definition of a *quarcle* and encourages related geometrical research. Furthermore we describe a simple software tool\(^1\) for *quarcle* construction and representation. The design of such a tool is part of our research in teaching programming.

![Figure 1: cG-simulation of a quarcle.](image)

\(^1\)Computer Geometry system CircleGraphics *cG*
We briefly recall the notion of different norms for $v = (x, y) \in \mathbb{R}^2$. In daily life, we are fine with the Euclidean Norm $\|v\|_2^2 := x^2 + y^2$. Alternatives are the Manhattan-norm $\|v\|_1 := |x| + |y|$ and the maximum-norm $\|v\|_0 := \max(|x|, |y|)$. In what follows we consider sets, so-called generalised circles, given by

$$B_{m,r}^{p} := \{v \in \mathbb{R}^2 \mid \|v - m\|_p \leq r\}, \quad p \in \{0, 1, 2\}, \ m = (m_x, m_y) \in \mathbb{R}^2, \ r > 0.$$  

This setting gives a unified view on the geometric objects relevant in this note. A circle in a common sense is obtained by choosing the Euclidean-, a quad by the maximum- and a rhombus by the Manhattan-norm. As will be seen below, this allows for a simplified implementation of an is element decision in our software-solution.

Announced as objects obtained by intersections and overlays of quads and circles, we state the

**Definition:**

- A quarcle $q^0$ of level 0 is a generalised circle $B_{m,r}^{p}$.
- A quarcle $q^n$ of level $n$ is a set $P_1(q^n)@P_2(q^n)$, with $@ \in \{\cap, \cup, \setminus\}$ and $P_i(q^n)$ quarcles of levels less than $n$ denoting the parents of $q^n$.

Using this notation the task given to students by Börstler & Sperber [1], compactly written, reads

**Statement of Task:** Design a Software tool deciding wether $v \in q^n$.

"Is $v$ Element of $q^n$?" may be answered in pseudo code with obvious meaning by the

**Algorithm:**

`IF $\|v - m\|_p \leq r$ THEN IsIn:=TRUE; ELSE IsIn:=FALSE; RETURN IsIn`

in the case $n=0$. For $n>0$ and $@ \in \{\cap, \cup, \setminus\}$ condition $\|v - m\|_p \leq r$ is simply replaced by

$$v \in P_1(q^n) \land v \in P_2(q^n), \quad v \in P_1(q^n) \lor v \in P_2(q^n), \quad v \in P_1(q^n) \land v /\in P_2(q^n).$$

**Remark:** Preliminaries, task and principal solution may be presented on one page, followed by two pages presenting implementation including explanation.
3 Programming

The underlying programming philosophy in this work is influenced by Job’s *simplicity is the ultimate sophistication* [2] and Wirth’s *plea for lean software* [8], where he took Einstein’s *Make it as simple as possible, but not simpler* as a mantra. Based on experiences since 1984 in Learning, Teaching [5] and Applying [4] programming languages - Pascal, C & C++, Java, Oberon - the choice in this note is *Oberon* [3, 6]. Implementations employ the Oxford Oberon Compiler written by M. Spivey.

In what follows, we present the design of a module providing the *is element decision* mentioned above. The head starts with

```pascal
MODULE Quarcles;
IMPORT (* modules for textual input *)
  U:=Utilities,F:=Files,S:=Strings,C:=Conv;
CONST
  Qmax = 64; (* max number of quarcles *)
  Zmax = 80; (* max length of textzeile *)
  inp = "quarcles.txt"; (* Name of quarcles code *)
```

As defined above, a quarcle of level 0 may be described by three REAL and one INTEGER value, representing midpoint (x,y), radius r and norm p. Otherwise, setting these values to 0, we need information how the parents \( P_1, P_2 \) can be addressed. A CHAR may indicate, how they are combined, e.g. \(+,\ast,-\) representing \(\{\cup,\cap,\setminus\}\). Consequently a quarcle in our implementation is given by

```pascal
TYPE Quarcle = POINTER TO QuarcleType;
QuarcleType = RECORD
  x,y,r : REAL; p : INTEGER;
  p1,p2 : Quarcle; op : CHAR
END;
```

Case distinction between level 0 or greater is done by \(r>0\) or \(r=0\). We assume that data of \( nq \) quarcles are listed in an ARRAY after reading a textfile.

```pascal
VAR nq : INTEGER; (* number of quarcles *)
  ql : ARRAY Qmax OF Quarcle; (* QuarcleList *)
```

After these preparations the *is element decision* can be formulated by

```pascal
PROCEDURE IsIn(q:Quarcle; x,y:REAL) : BOOLEAN;
VAR R : BOOLEAN;
BEGIN
  IF q.r>0. THEN
    R := IsInCircle(q,x,y)
  ELSE
    R := IsInMixed(q,x,y)
END;
```

2www.oberon07.com, loaded 11.4.2020
where the procedures \texttt{IsInCircle}(q,x,y) and \texttt{IsInMixed} are implemented as follows

\begin{verbatim}
PROCEDURE IsInCircle(q:Quarcle; x,y:REAL) : BOOLEAN;
VAR R : BOOLEAN; d,dx,dy,r : REAL;
BEGIN
  dx := q.x-x; dx := ABS(dx); (* basic dist *)
  dy := q.y-y; dy := ABS(dy); (* information *)
  r := q.r;
  IF (dx>dy) THEN d := dx ELSE d := dy; (* max norm *)
  IF q.p = 1 THEN d := dx + dy; END; (* manhattan *)
  IF q.p = 2 THEN d := dx*dx + dy*dy; r := r*r; END; (* euclidean *)
  IF d<=r THEN R := TRUE; ELSE R := FALSE; END;
RETURN R
END IsInCircle;

PROCEDURE IsInMixed (q:Quarcle; x,y:REAL) : BOOLEAN;
VAR R : BOOLEAN; p1,p2 : Quarcle;
BEGIN p1 := q.p1; p2 := q.p2; R := FALSE;
  IF q.op = "+" THEN IF IsIn(p1,x,y) OR IsIn(p2,x,y) THEN R := TRUE; END;
  ELSIF q.op = "+" THEN IF IsIn(p1,x,y) & IsIn(p2,x,y) THEN R := TRUE; END;
  ELSIF q.op = "-" THEN IF IsIn(p1,x,y) & ~IsIn(p2,x,y) THEN R := TRUE; END;
  ELSE RETURN R
END IsInMixed;
\end{verbatim}

Eventually a procedure is exported, enabling a client to decide whether a given point is element of the \textit{i}th quarcle in a given list.

\begin{verbatim}
PROCEDURE Check*(i : INTEGER; x,y : REAL) : BOOLEAN;
BEGIN DEC(i); RETURN IsIn(ql[i],x,y)
END Check;
\end{verbatim}

The module closes with a procedure - details omitted here - reading and converting the textual input file and its invocation in module’s body.

\begin{verbatim}
PROCEDURE Init(); (* quarcle description from file to program *)
BEGIN Init;
END Quarcles.
\end{verbatim}

An example of standard input file \texttt{quarcles.txt} is

\begin{verbatim}
440. 280. 200. 1 Diamond
300. 25. 200. 0 Quad
1 * 2
1 + 2
0. 200. 200. 2 Circle
4 + 5
6 - 2
\end{verbatim}

organised as described above.
4 Graphical simulation

Now, we present the implementation of our graphic tool cG, generating a virtual window. In more detail, each point in an rectangular area is checked to be element of a given quarcle. Result is a textfile containing 0 or 1 information for each point. This format can easily be visualised e.g. with gnuplot.

```pascal
MODULE CircleGraphics;
    IMPORT Args, Conv, F:=Files, Q:=Quarcles;
    CONST
        Width = 640; Height = 480; (* for graphical output *)
        out = "cG.dat"; (* for data output *)
    VAR
        VW : ARRAY Height+1 OF ARRAY Width+1 OF BOOLEAN; (* Virtual Window *)
        x, y : REAL; i, j, v, q : INTEGER;
        s : ARRAY 20 OF CHAR; fp : F.File;
    BEGIN q := 1;
        IF Args.argc>=2 THEN Args.GetArg(1,s); q := Conv.IntVal(s); END;
        x := 0.; y := 0.;
        FOR i:=0 TO Width DO x := FLT(i);
            FOR j:=0 TO Height DO y := FLT(j); VW[j][i] := Q.Check(q,x,y); END;
            END;
        fp := F.Open(out, "w");
        FOR i:=0 TO Height DO
            FOR j:=0 TO Width DO
                IF VW[i][j] THEN v:=0; ELSE v:=1; END; F.WriteInt(fp,v,2);
                END;
            F.WriteLn(fp);
        END;
    END CircleGraphics.

Using the input file `quarcles.txt`, quarcle 7 is shown in Figure 1.

References


    https://doi.org/10.12988/imf.2007.07001


Received: May 19, 2020; Published: June 24, 2020