Technische Universität Berlin
Institut für Mathematik
Course: Mathematical Visualization I WS12/13
Professor: John M. Sullivan
Assistant: Charles Gunn

18.04.2013

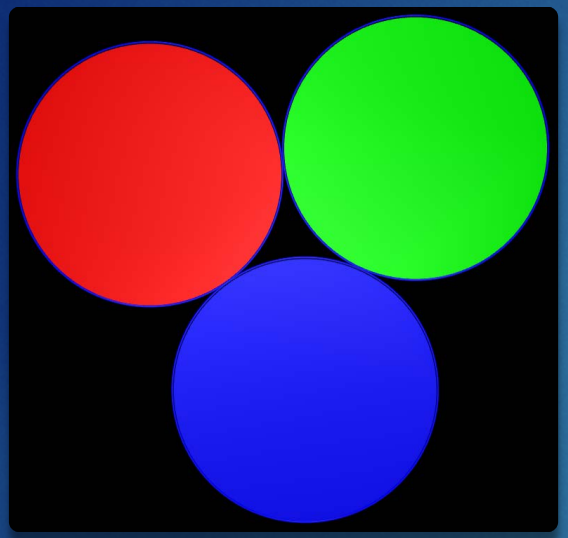
The Apollonian Gasket

by Eike Steinert & Peter Strümpel

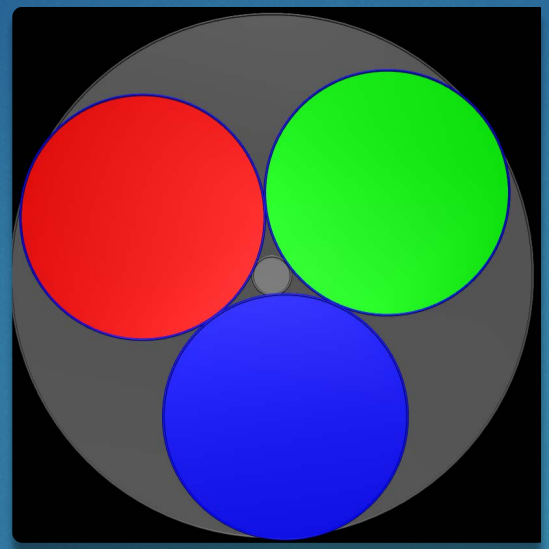
structure

1. Introduction
2. Apollonian Problem
    1. Mathematical background
    2. Implementation
    3. GUI
3. Apollonian Gasket
    1. Mathematical background
    2. Implementation
    3. GUI
4. Future prospects

Introduction – Apollonian Gasket

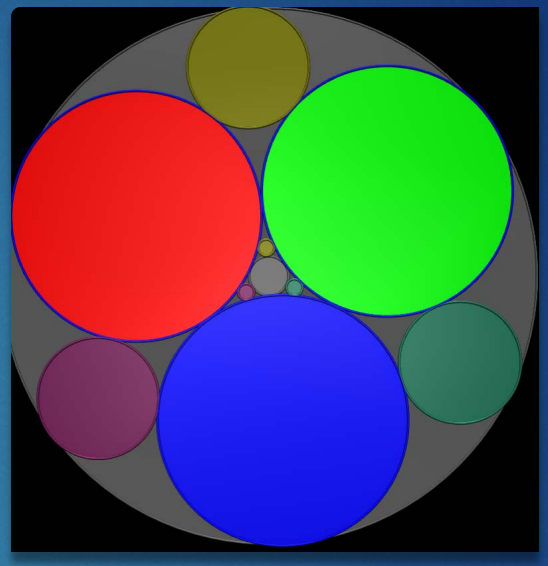


- It is generated from **triples** of circles
- Each cirlce is **tangent** to the other two



construct the two **Apollonian circles** which touches the given ones:

- internally and
- externally

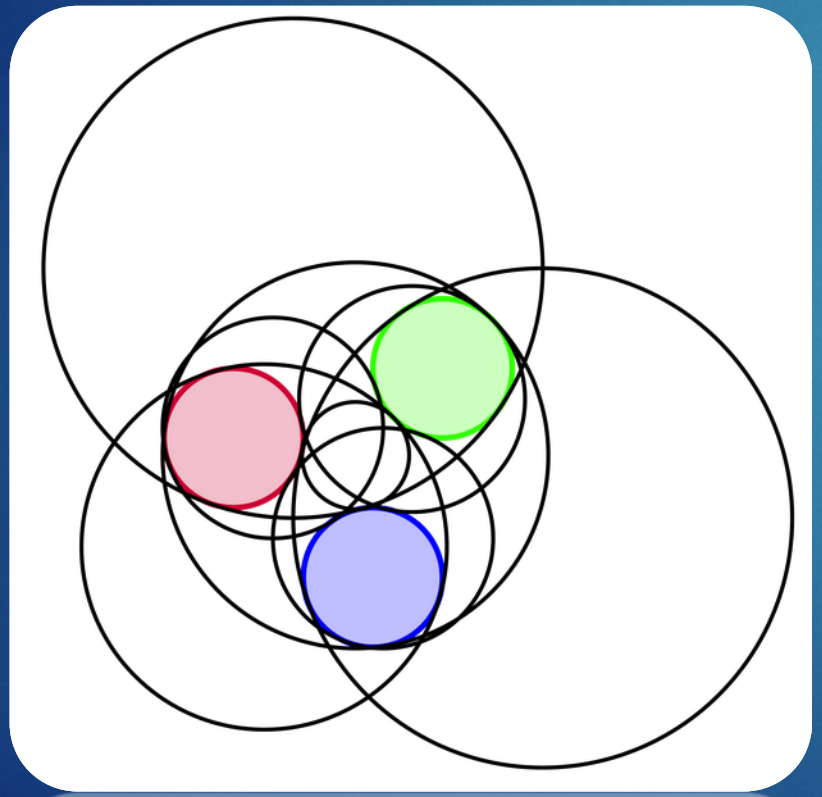


Take again 3 tangent circles

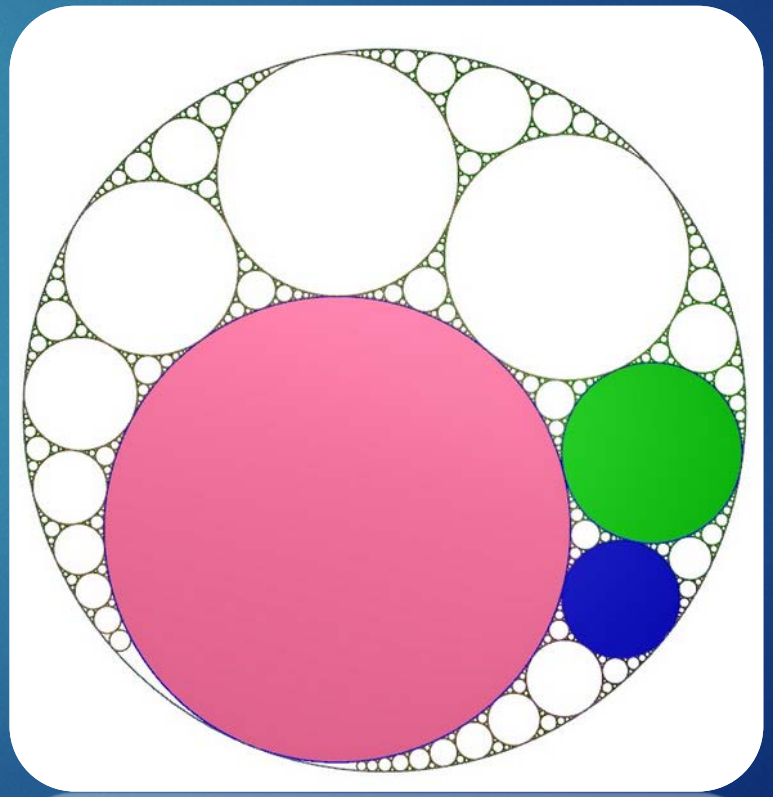
Constuct again cirlces which touches the given ones

Calculation of the Apollinian circles

Apollonian Problem



Apollonian Gasket



Apollonian problem - mathematical background

- Apollonius of Perga ca. 200 b.c.: how to find a circle which touches 3 given objects?
- objects: lines, points or circles
- limitation of the problem: only circles
- first who found algebraic solution was Euler in the end of the 18. century

Apollonian problem - mathematical background

- algebraic solution based on the fact, that distance between centers equal with the sum of the radii
- so we get system of equations:

$$(x - x_1)^2 + (y - y_1)^2 - (r \pm r_1)^2 = 0$$
$$(x - x_2)^2 + (y - y_2)^2 - (r \pm r_2)^2 = 0$$
$$(x - x_3)^2 + (y - y_3)^2 - (r \pm r_3)^2 = 0$$

- +/- determines external/internal tangency
- 2^3=8 combinations => 8 possible circles

Apollonian problem - mathematical background

- subtracting
  - two linear equations:

$$ax + by + cr = d$$

$$a'x + b'y + c'r = d'$$

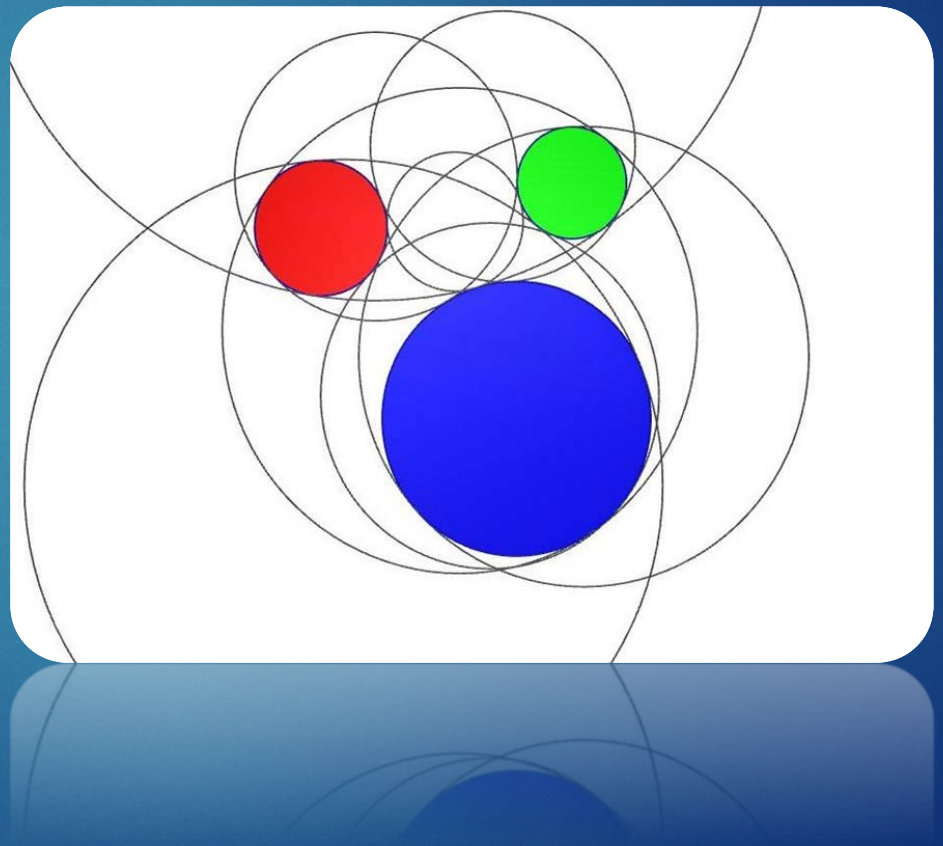
- solving with respect to **r** we get:

$$x = \frac{b'd - bd' + bc'r - b'cr}{ab' - a'b}$$

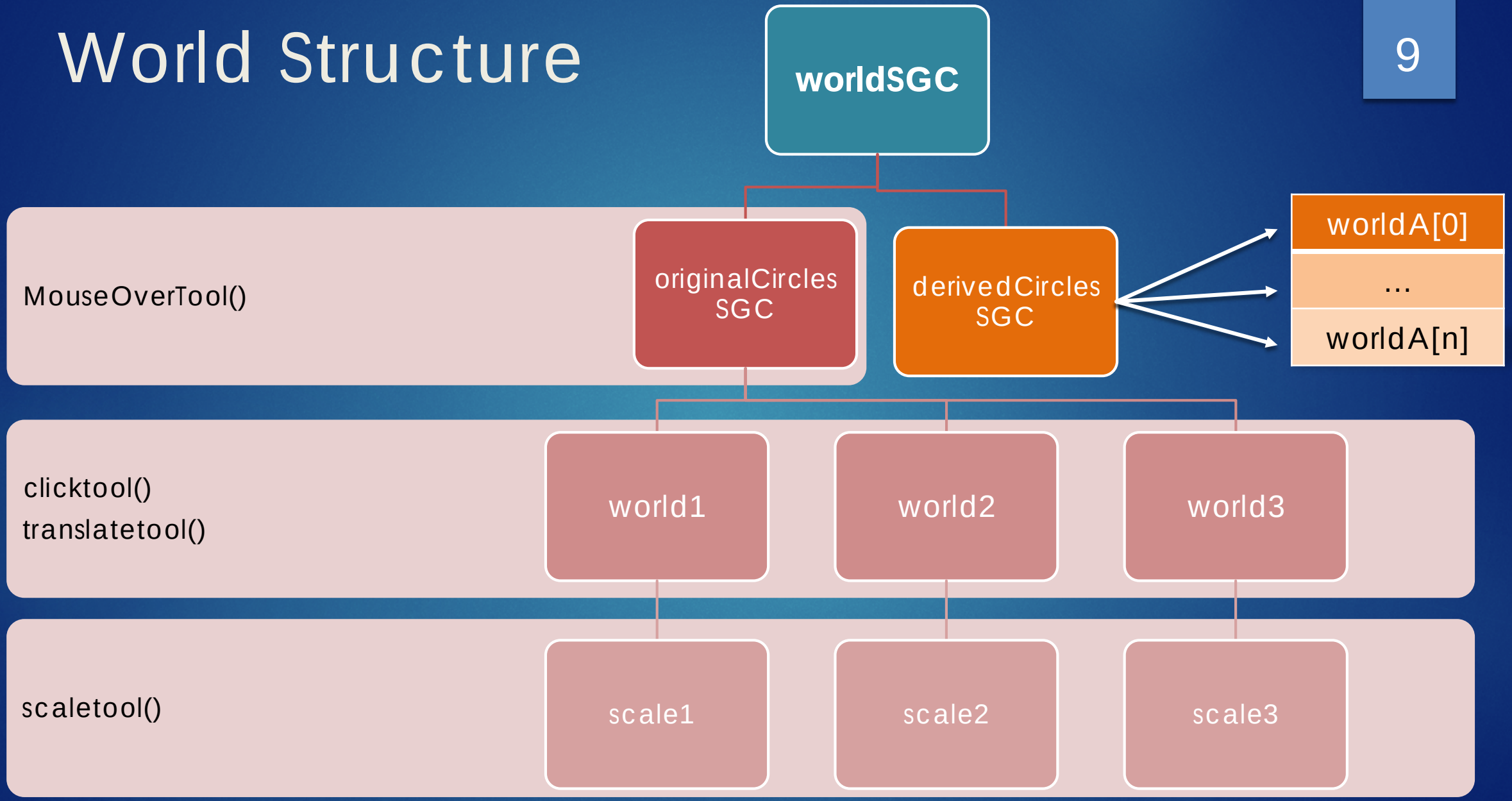
$$y = \frac{ad' - a'd + a'cr - ac'r}{ab' - a'b}$$

Apollonian problem - mathematical background

- these results in first equation quadratic expression for r
- mostly only one r is real
- so, we get all 8 solution circles (if they exist)



World Structure



Apollonian problem - Tools

- `MouseOverTool()` - highlights the geometry which is "hit" by the pointer device, by changing its color
- `TranslateTool()` - moving around the original Cirlces

- `scaletool()`
  - mouse wheel up → scale up
  - mouse wheel down → scale down

Apollonian problem - Implementation

- Circles - constructed by regular polygons (*using de.jreality.geometry.Primitives*)
- Class `Circle` - for handling information about coordinates, radius and color of a circle

```
Circle(x-value, y-value, radius, red, green, blue)
```

- method

```
updateGasket(Circle c1, Circle c2, Circle c3, io1, io2, io3)
```

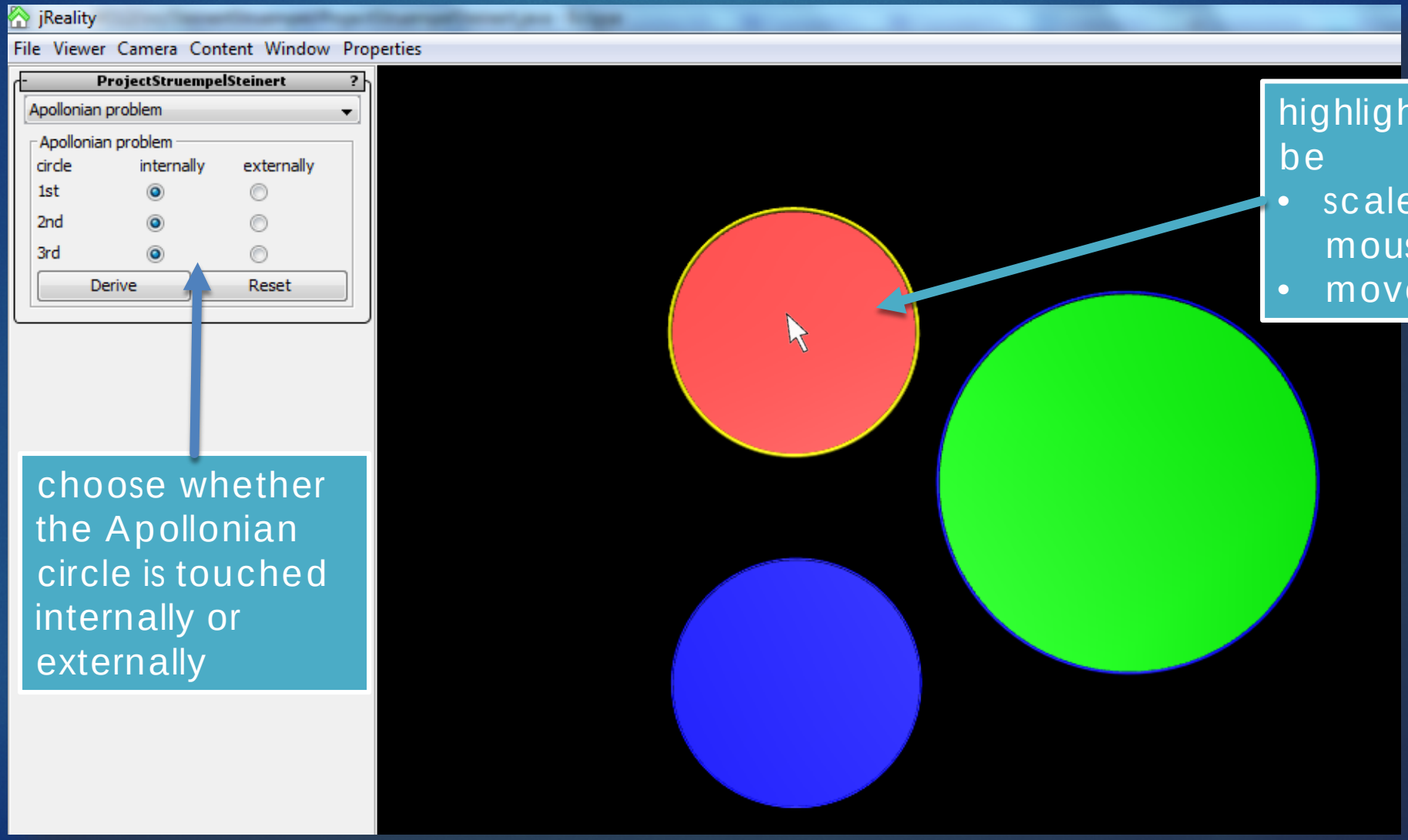
constructs the Apollonian circle as *indexedLineSetFactories*

inner or outer desired tangency

1 → internally

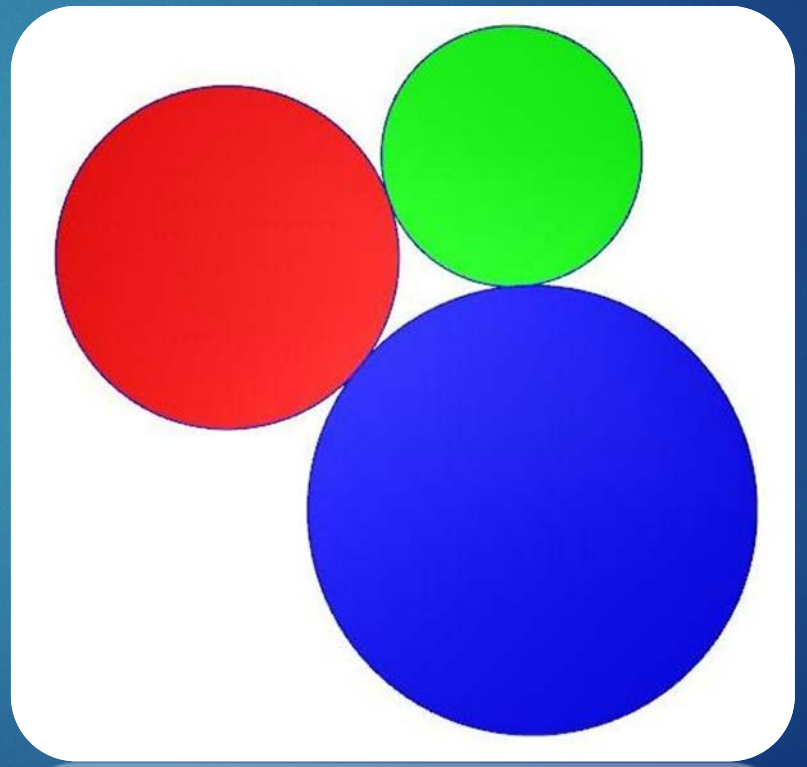
-1 → externally

GUI – Apollonian problem



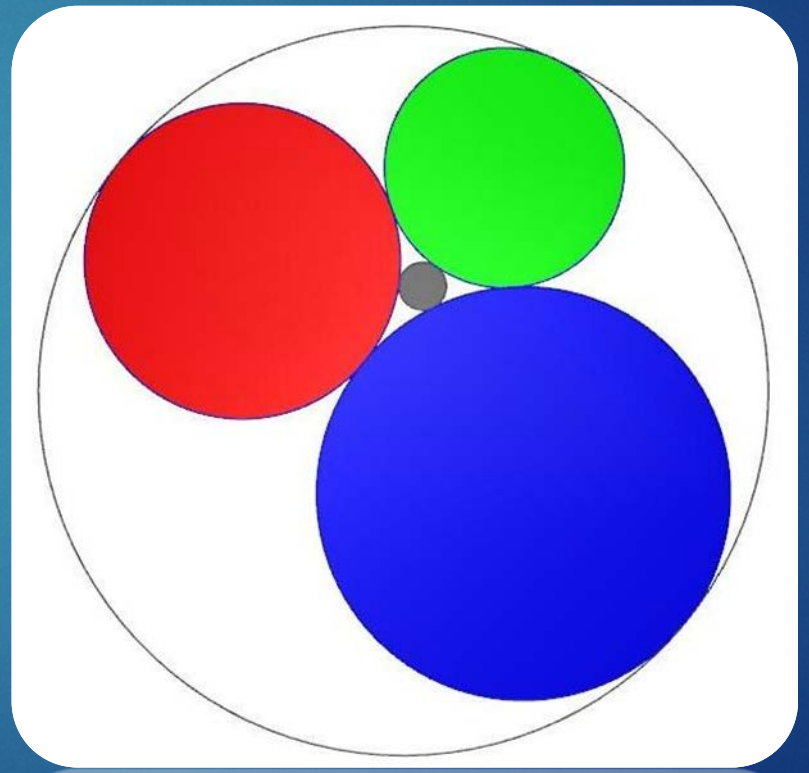
Apollonian gasket - mathematical background

- fractal invented from Apollonius of Perga, too
- also called Apollonian net
- start-setting:
  - three circles - a triple
  - all tangent to each other



Apollonian gasket - mathematical background

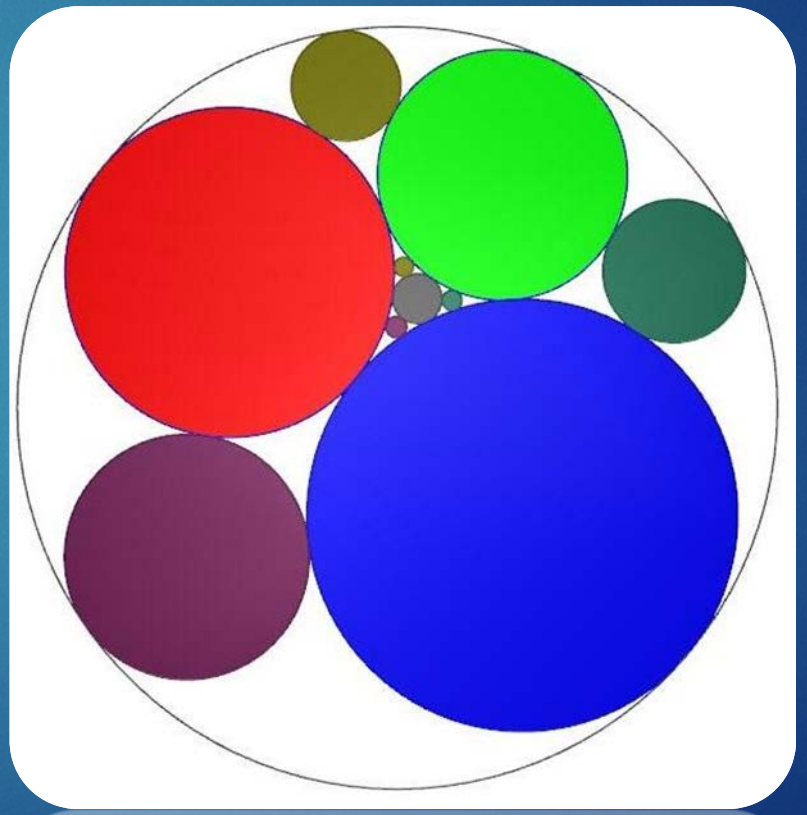
- the Apollonian problem for this triple has only two solutions
- other possible solutions are congruent with the given circles
- now we got 5 circles



Apollonian gasket - mathematical background

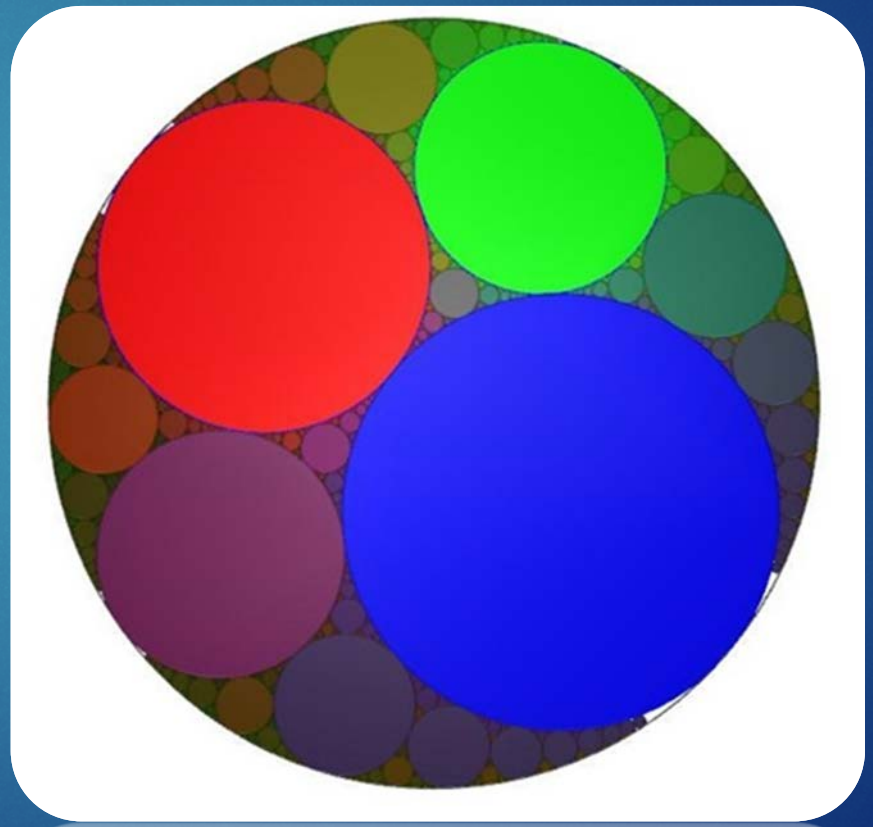
- 6 new tripel of tangent circles

- Apollonian Problem has for every
  tripel one new solution circle
- now we got 6 more circles
- iterations can go on and on



Apollonian gasket - mathematical background

- 3 new circles for every new circle in the next iteration
- in the 9h step the amount is already $2*3^9=39366$ circles
- the Apollonian circles can be calculated recursively



Apollonian gasket - mathematical background

- for 3 tangent circles (c_1, c_2, c_3) (after 1st step) find the solution s_1 of the Apollonian problem
- now find the solution s_2 for (c_1, c_2, s_1), s_3 for (c_1, c_2, s_2)
  ...
- Doing the same for (c_1, c_3, s_1) and (c_2, c_3, s_1) constructs recursively the gasket

Apollonian gasket - software implementation

- in our program we differ between the "inner" circles and the "outer„

```
void childChainsForInner(Circle a, Circle b, Circle c, int n) {
     if (n>0) {
     Circle d=updateGasket(a,b,c,-1,-1,-1);
     childChainsForInner(a,b,d,n -1);
     childChainsForInner(a,c,d,n-1);
     childChainsForInner(b,c,d,n-1);
     }
    }
```

n – number of iterations

will be +1 in the chain for "outer"

Apollonian gasket – Implementation

- `updateGeometry()` - makes one circle tangenet to an other one, by moving the circle

- `checkTangency()` - changes color of the orig. circles depending on the tangency
  - white - not tangent
  - colored - tangent

- `childChainsForOuter()/ChildChainsForInner()`
  - `in comibantion with updateGasket()`

construct the new circles, if not already exist
checked by method `circleExists()`

Apollonian gasket - software implementation

- we handle the colors with

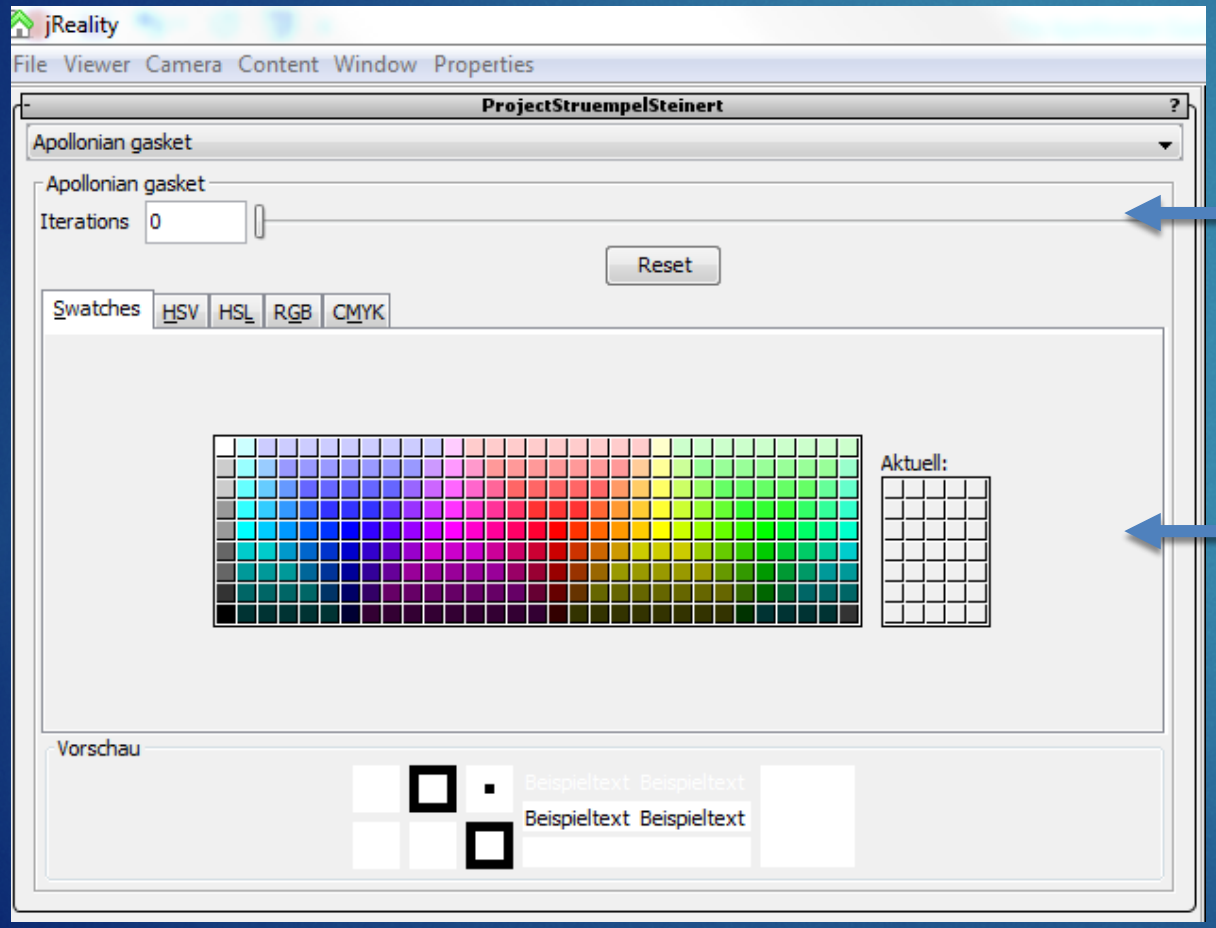
```
new Color(int red, int green, int blue)
```

- new derived circles have mixture of the triple
- when calculating the Apollonian circle

```
ared = (c1.red + c2.red + c3.red)/3
```

- same for green & blue gives the proper mixture

GUI– Apollonian problem



constructs & determinates the depth of the appolnian gasket

changes the color of the last picked circle

future prospects

- avoiding circle positions which can not produce gaskets
- color-mix dependent on radii
- Automatic tangency-sticking
- producing a Apollonian Gasket with spheres
(Appollonian packing)