## Part A: System constant declaration

PI = 3.14159;

from math import \*

anyValue = pi

## Part B: System function declaration

from sympy.geometry import \*

function print(info:int const, ...):void;

print 1

print 1+1

function print(info:float const, ...):void;

print 1.1

print pi

function print(info:char const, ...):void;

print ‘a’

function print(info:bool const, ...):void;

print True

print False

function print(info:string const, ...):void;

print “hello world”

print ‘hello world2’

function int\_to\_string(input:int):string;

str(input)

function float\_to\_string(input:float):string;

str(input)

function bool\_to\_string(input:bool):string;

str(input)

function char\_to\_string(input:char):string;

str(input)

## Part C: Geometric function declaration

from sympy.geometry import \*

1. Dot

//geometricShape submodel: dot

//dot parameter name: 'x' 'y'

model dot:

topmodel(geometricShape);

step[2]:float;

stepSet[2]:bool;

dot(x:float, y:float);

pt = Point(3.3, 4.4)

function getX():float;

pt.x

function getY():float;

pt.y

function distance(dot1:dot const):float;

pt2 = Point(5.5, 6.6)

pt.distance(pt2) // NOTE: could be symbolic expr 11\*sqrt(2)/5

float(pt.distance(pt2)) // returns a float

function distance(line1:line const):float;

line1.distance(pt1)

function setRunstep(val:float,pos:char):void;

function getRunstep(pos:char const):float;

end

1. Line

//geometricShape submodel: line

//line parameter name: 'a' 'b'

//line formula y = ax + b

model line:

topmodel(geometricShape);

step[2]:float;

stepSet[2]:bool;

endPoint[2]:dot;

endPointset:bool;

a:float;

b:float;

line(a:float,b:float);

line1 = Line(Point(0, b), Point(1, a+b)) // because only Line(pt1, pt2)

line(dot1:dot,dot2:dot);

line2 = Line(dot1, dot2)

line(a:float,b:float,endpointx1:float,endpointx2:float);

line3 = Segment(Point(endpointx1, a \* endpointx1 + b), Point(endspointx2, a \* endpointx2 + b)) // only Segment(pt1, pt2)

line(dot1:dot,dot2:dot,endpointx1:float,endpointx2:float);

line4 = Segment(Point(endpointx1, (dot2.y – dot1.y) / (dot2.x – dot1.x) \* (endpointx1 – dot2.x) + dot2.y), Point(endpointx2, (dot2.y – dot1.y) / (dot2.x – dot1.x) \* (endpointx2 – dot2.x) + dot2.y))

function getPara(pos:char const):float;

if (pos == ‘a’):

return line1.slope

else:

return -line1.coefficients[2] / line1.coefficients[1]

// line.coefficients returns (a,b,c) where ax + by + c

function getY(x:float const):float; //Exception may occur.

// What about a segment out of bound?

return (line.p2.y – line.p1.y) / (line.p2.x – line.p1.x) \* (x – line.p2.x) + line.p2.y

function getMidpoint():dot;

//If endPointset = false, return [0,0]?

return line.midpoint()

function setRunstep(val:float,pos:char):void;

function getRunstep(pos:char char):float;

function intersect(polygon1:polygon const):float[];

// Polygons

function intersect(circle1:circle const):float[];

if (line.distance(circle.center) < circle.radius):

return line.intersection(circle1)[1].distance(line.intersection(circle1)[0])

else:

return 0

end

1. Circle

//geometricShape submodel: circle

//circle parameter name: 'a' 'b' 'r'

//circle formula r^2 = (x-a)^2 + (y-b)^2

model circle:

topmodel(geometricShape);

step[3]:float;

stepSet[3]:bool;

a:float;

b:float;

r:float;

circle(center:dot,radius:float);

circle = Circle(dot, radius)

function setRunstep(val:float,pos:char):void;

function getRunstep(pos:char):float;

function getCenter():dot;

circle.center

function getRadius():float;

circle.radius

//Out\_of\_range Exception may occur.

function getY(x:float const):float[];

if (abs(x - circle.center.x) > circle.radius):

raise ValueError('A very specific bad thing happened')

else:

print [circle.center.y +

sqrt(circle.radius \* circle.radius - (x - circle.center.x) \* (x - circle.center.x)),

circle.center.y -

sqrt(circle.radius \* circle.radius - (x - circle.center.x) \* (x - circle.center.x))]

function intersect(polygon1:polygon const):float[];

// POLYGONS

function intersect(circle1:circle const):float[];

circle1.intersection(circle)[1].distance(circle1.intersection(circle)[0])

end

1. Polygon

//geometricShape submodel: polygons

//polygons parameter name: 'a' 'b' ...

//polygons formula: A set of dots

model polygons:

topmodel(geometricShape);

step[]:float;

stepSet[]:bool;

apexs[]:dot;

polygons(num\_of\_apex:int,apex[]:dot);

// unwrapping a dot array is very troublesome! Change to multiple dots?

// Delete num\_of\_apex parameter?

// I assumed regular polygon here

poly = RegularPolygon(center, distanceFromCenterToVertex, numOfSides)

function setRunstep(val:float,pos:char):void;

function getRunstep(pos:char):float;

function getCenter():dot;

// But how do you define a center for an irregular polygon?

poly.center

function getRadius():float;

// Here for radius too?

poly.radius // distance from center to vertex

//Out\_of\_range Exception may occur.

function getY(x:float const):float[];

// calculate

function intersect(polygon1:polygon):float[];

poly.intersection(polygon1)[1].distance(poly.intersection(polygon1)[0])

end

1. Runset

//controltype runset

model runset:

runEnable:bool;

times\_of\_run:int;

shape[]:geometricShape;

runpara[]:char;

runset(times\_of\_run:int, g1:geometricShape, run\_para\_g1:char, ...);

function refresh():void;

function addElement():bool;

function removeElement(g:geometricShape,para:char):bool;

function enableRun():void;

function disableRun():void;

end