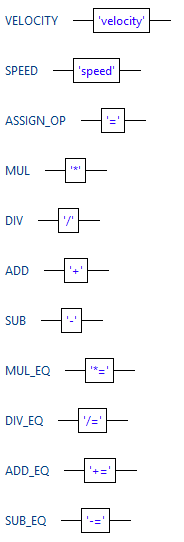
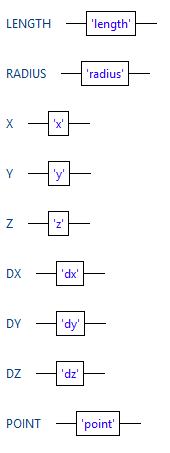
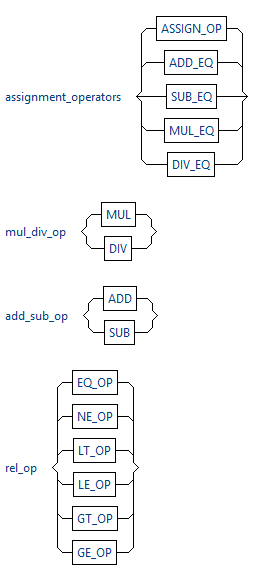
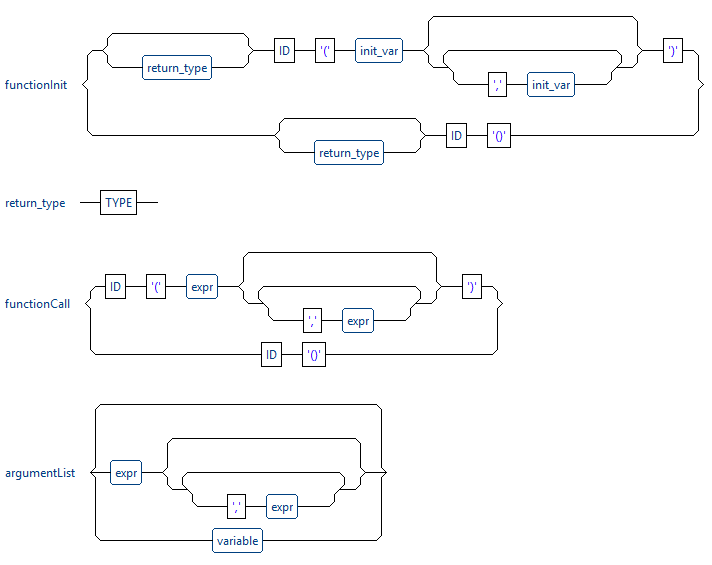
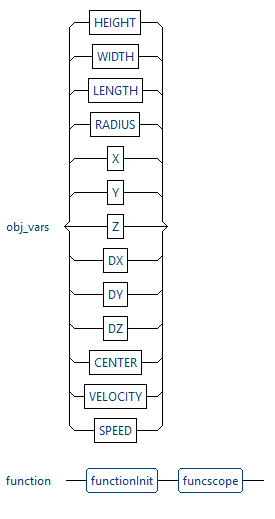
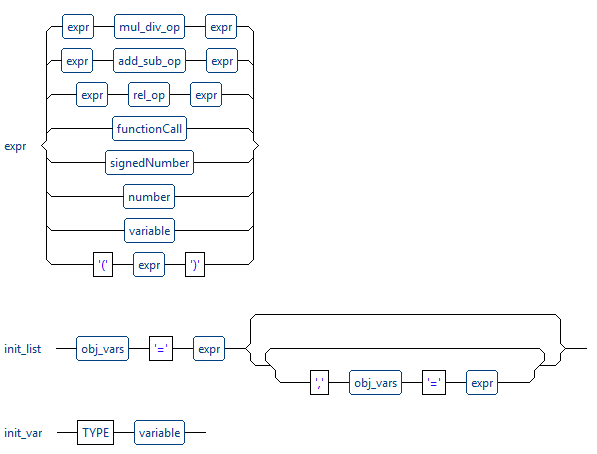
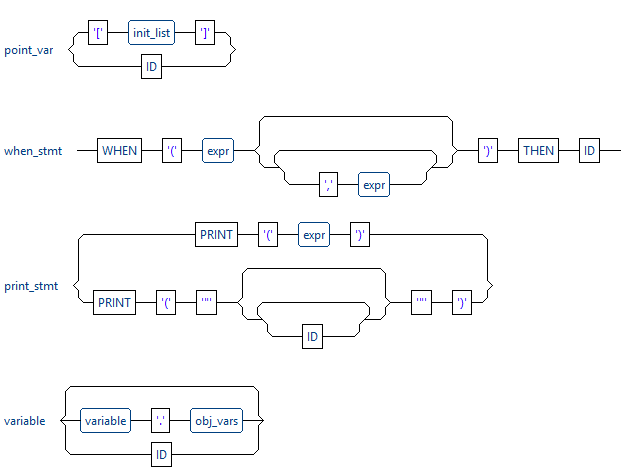
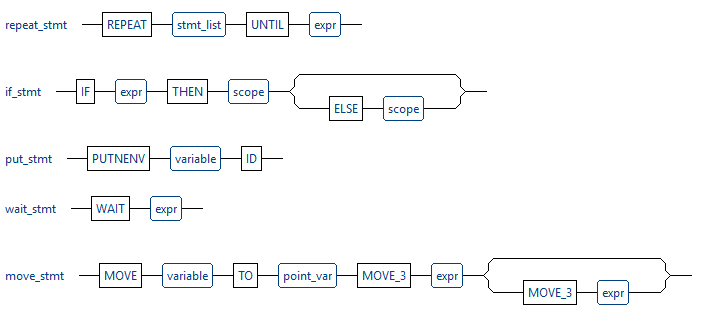
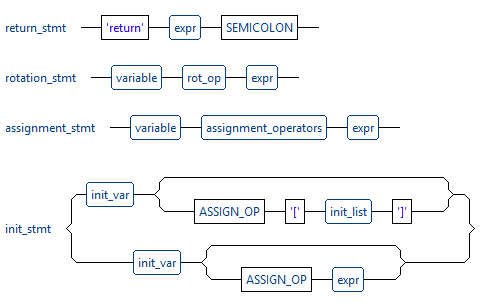
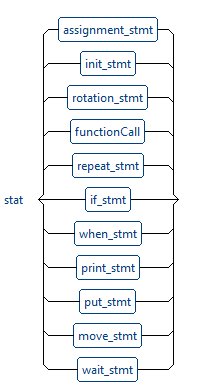
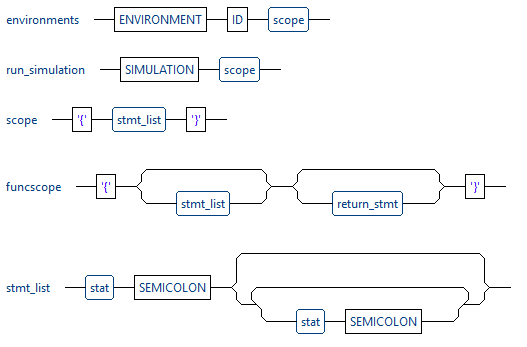
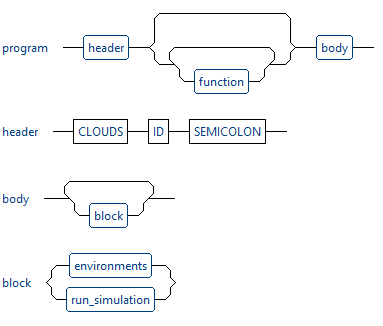
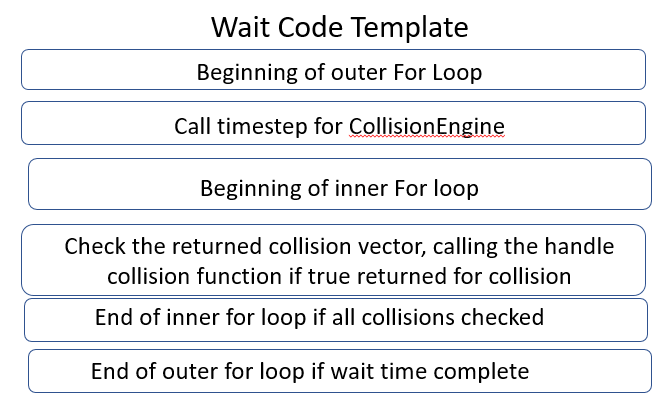
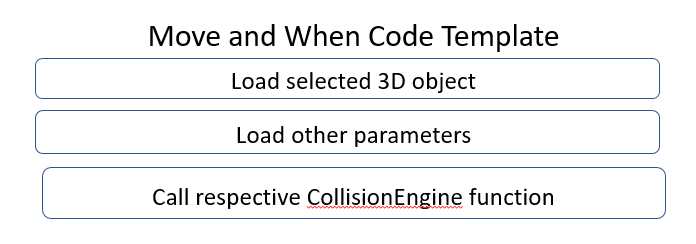
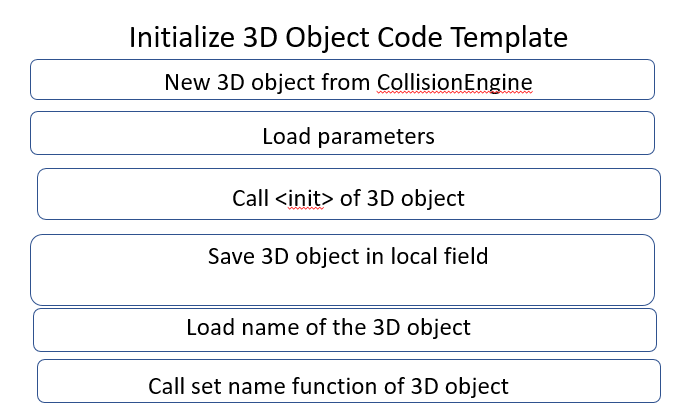
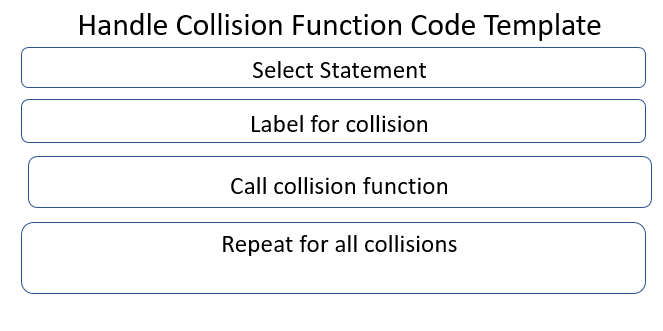
Clouds Written report

* + Syntax diagrams for key source language constructs.



* + Code templates for key source language constructs.



* + Instructions on how to build your compiler(with cmake).

Prerequisites:

antlr4 libraries are in /usr/local/lib

antlr4 header files are in /usr/local/include

If antlr4 files needed to be generated, use Clouds.g4 in src

To build compiler using cmake on command line(after cpp files are generated):

# run inside FinalCompilerDeliverables

mkdir build

cd build

cmake ..

make -j

cd ..

* + Instructions on how to run your compiler (script).

# To run compiler if executable is in FinalCompilerDeliverables:

./Clouds sample\_programs/Assignment7.clouds

java -jar jasmin.jar collisionengine/assignmentseven.j

javac collisionengine/\*.java

java -cp .:PascalRTL.jar collisionengine/assignmentseven

* Sample source programs written in your language to compile and execute.
  + Also in sample programs folder under Assignment7.clouds

Clouds assignmentseven;

/\*

Clouds Example Program

\*/

int functionTwo (int j)

{

return 2\*j;

}

environment environmentObject

{

int timetomovecube = functionTwo(5);

point pointObjectOne = [ x = 0, y = 0, z = 0 ];

point pointObjectTwo = [ x = 10, y = 0, z = 0 ];

point pointObjectThree = [ x = 10, y = 0, z = 25 ];

cube cubeobject = [height=1, width=1, length=2];

sphere sphereobject = [radius = 2];

cube cubeobjectTwo = [height=3, width=3, length=4];

putnenv cubeobject pointObjectOne;

putnenv sphereobject pointObjectTwo;

putnenv cubeobjectTwo pointObjectThree;

WHEN(cubeobject, sphereobject) THEN collisionOne;

WHEN(cubeobject, cubeobjectTwo) THEN collisionTwo;

WHEN(sphereobject, cubeobjectTwo) THEN collisionThree;

}

collisionOne(){

print("col one");

cubeobject.dy = 20;

sphereobject.dx = 5;

}

collisionTwo(){

print("col two");

cubeobject.dx = 10;

cubeobjectTwo.dz = 2;

}

collisionThree(){

print("col three");

sphereobject.dx = 8;

cubeobjectTwo.dz = 3;

}

simulation

{

move cubeobject to pointObjectTwo in timetomovecube;

wait 3;

}

* Sample output from executing your source programs.

col one

col two

col one

col two

col one

col two

Size of Environment: x = 100, y = 100, z = 100

Center of Environment: x = 50, y = 50, z = 50

Current Object Status:

cubeobject center is at: x=21, y=40, z=0 and velocity is: dx=10, dy=20, dz=0

sphereobject center is at: x=20, y=0, z=0 and velocity is: dx=5, dy=0, dz=0

cubeobjectTwo center is at: x=10, y=0, z=29 and velocity is: dx=0, dy=0, dz=2

Collisions being handled:

Number of Collisions = 3

0.12 seconds total execution time.