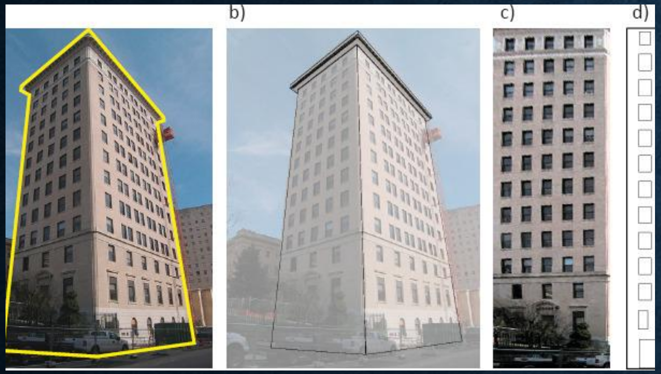
PROCEDURAL GENERATION OF BUILDINGS

CS 334

Colin Vinarcik

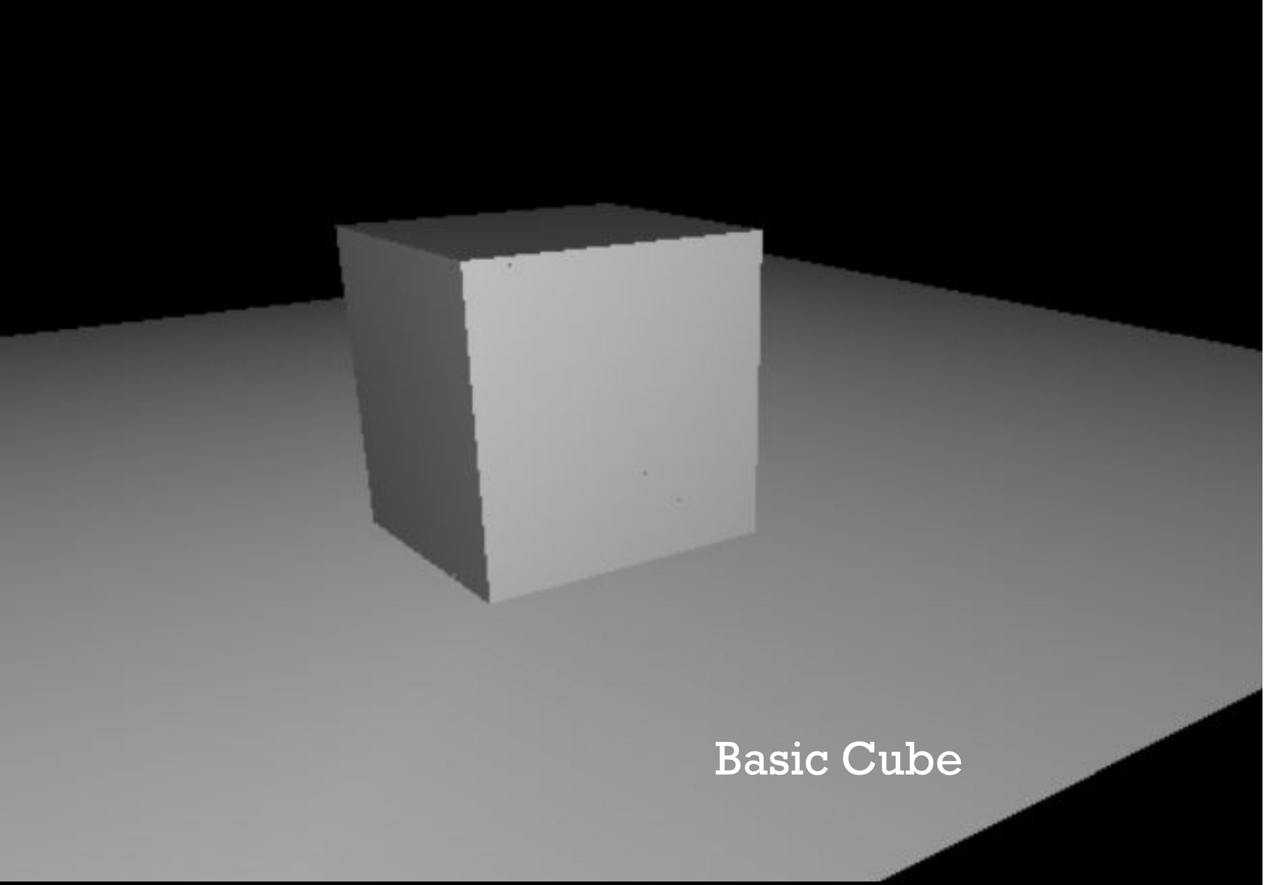
A LITTLE BACKGROUND

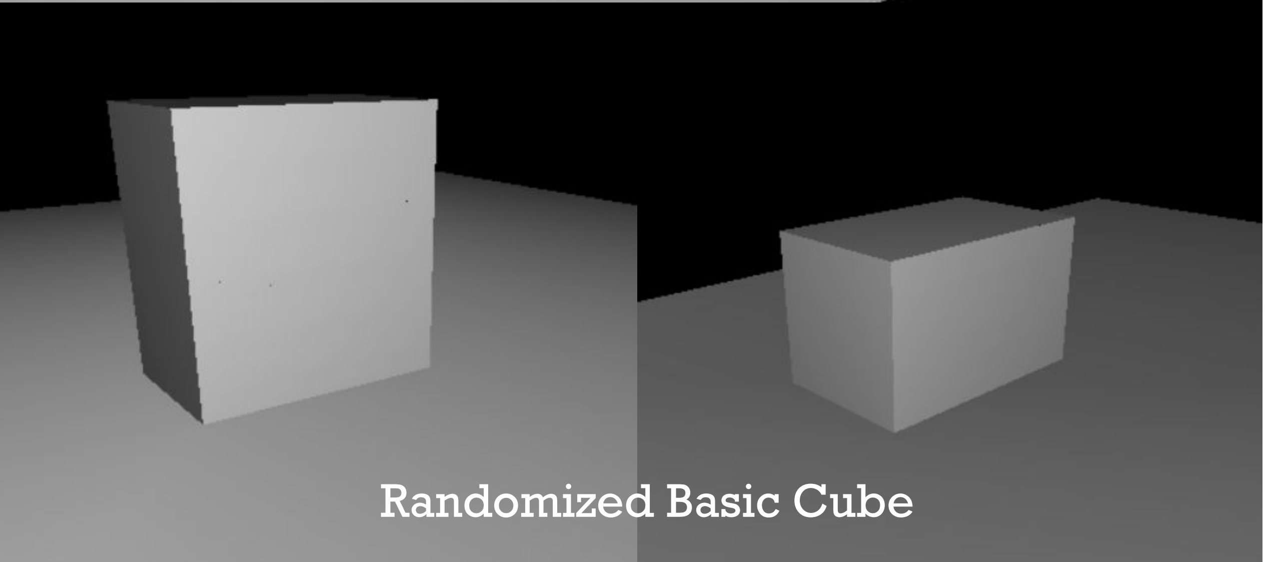


- Goal – make a building from a file based on modify Parameters or leave random: Building height, Building Angle, Windows, floor area.
- Use a basic sequential grammars, general rules to add, scale, translate, and rotate shapes.
- Basic split rule: The basic split rule splits the current scope along one axis. (Used to Design Floors and Building Features).

BASIC GRAMMAR

- Scope (x;y;z) - set the scope of a new geometry object
- Divide(x,y,z)[object,object] – divide the object and give the set the outcome to differet variables
- Translate(x,y,z) – used to translate an object somewhere in the world space
- Repeat(x,y) [object] – given a object repeat last command with variables for given number of times
- GeometryDraw(objectType)[image] – draw the give object, and use give image if it exists



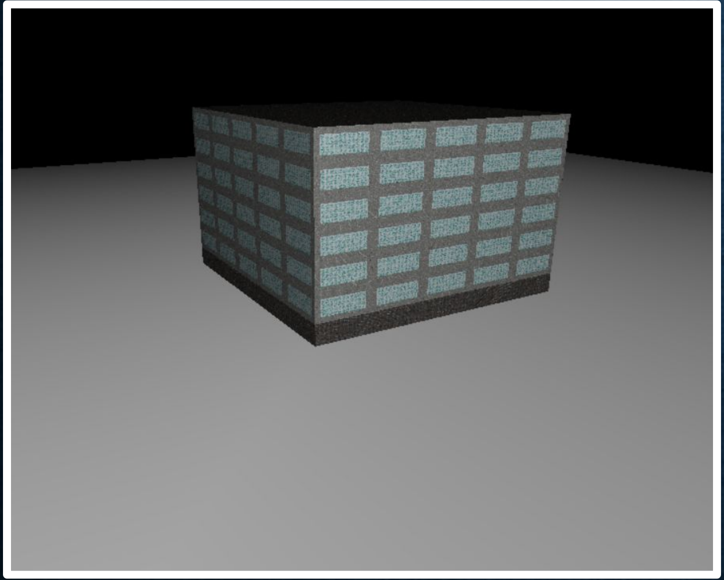


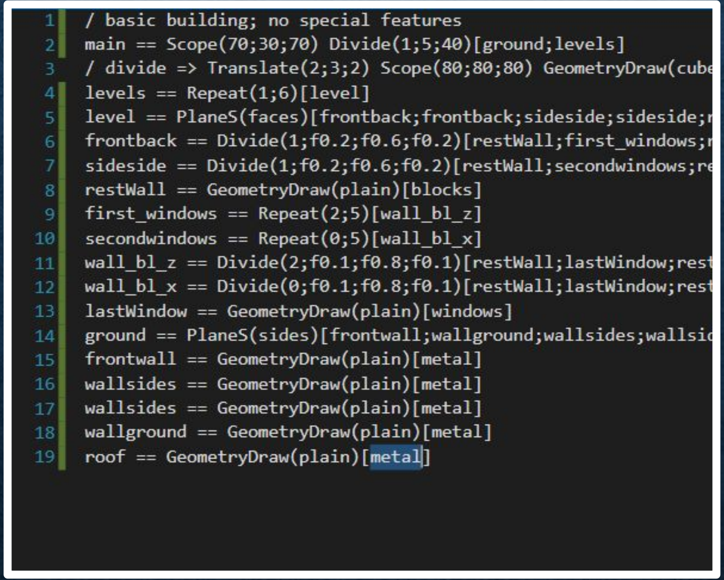
FIRST STEP

- Make a Parser to read a grammer from an input file.
- Use the give rule parsed in to generate a 3d world (first with no textures on objects)
- Add randomization of the rules to make a diverse set of possible buildings

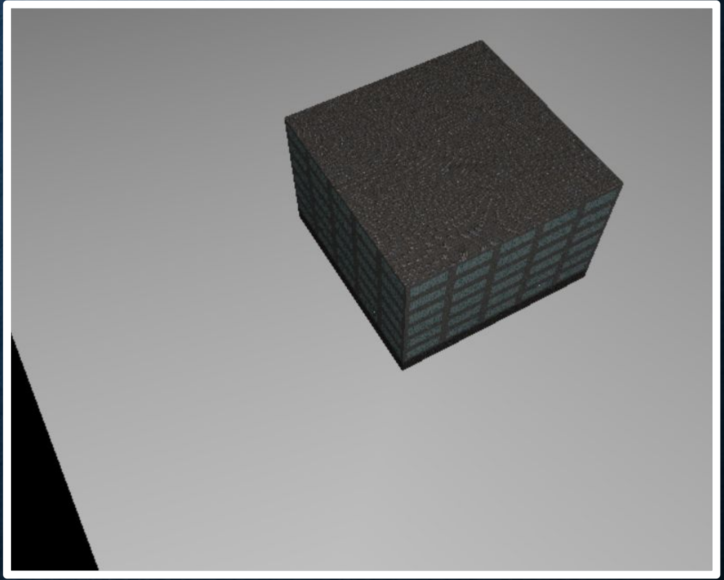
ADDITION OF TEXTURES

- Implanting Textures, and spiting of objects
- Allows for different parts of the building to have different textures, such as adding windows

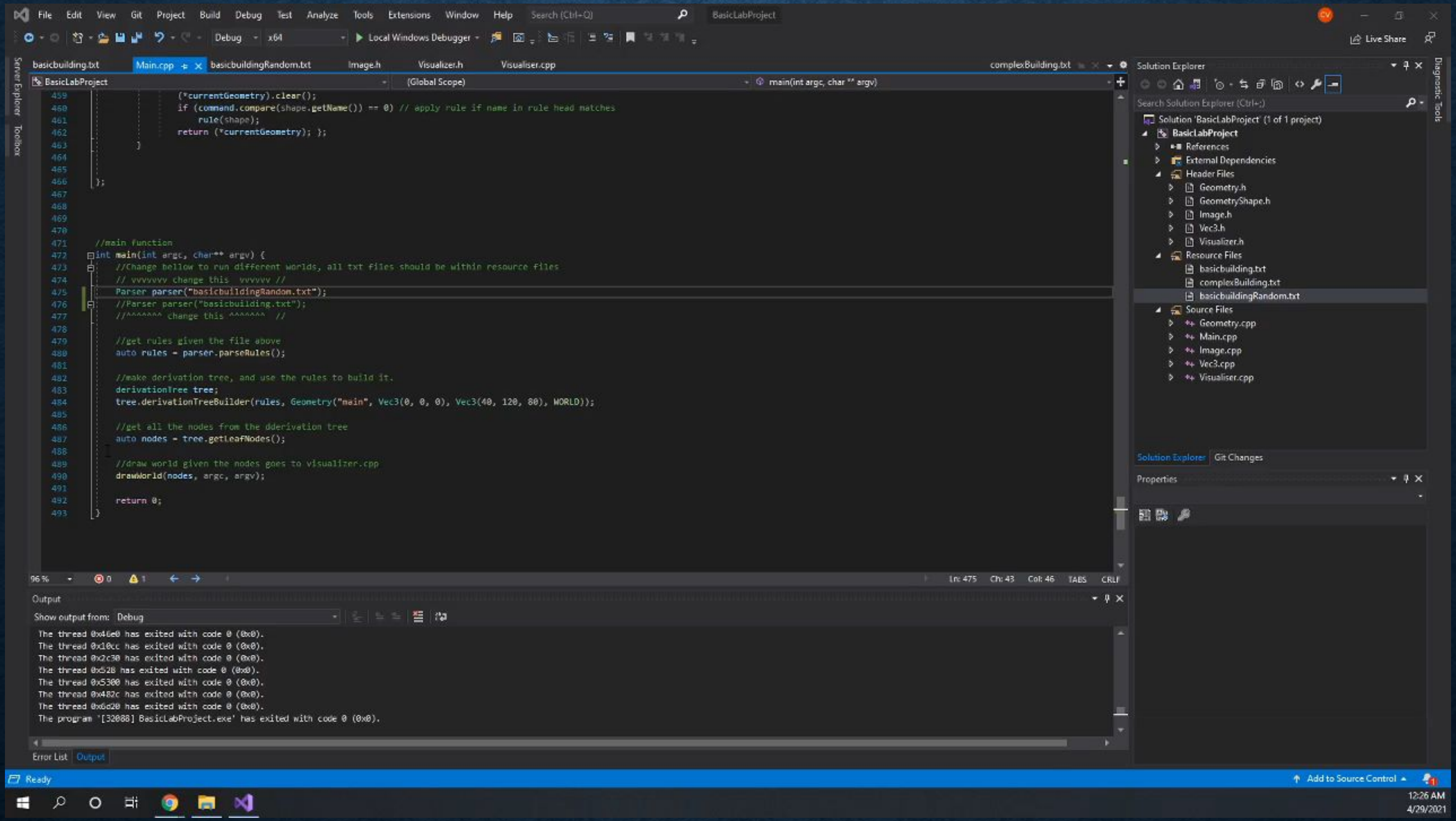




```
/ basic building; no special features
main == Scope(70;30;70) Divide(1;5;40)[ground;levels]
/ divide => Translate(2;3;2) Scope(80;80;80) GeometryDraw(cub
levels == Repeat(1;6)[level]
level == PlaneS(faces)[frontback;frontback;sideside;sideside;
frontback == Divide(1;f0.2;f0.6;f0.2)[restWall;first_windows;
sideside == Divide(1;f0.2;f0.6;f0.2)[restWall;secondwindows;re
restWall == GeometryDraw(plain)[blocks]
first_windows == Repeat(2;5)[wall_bl_z]
secondwindows == Repeat(0;5)[wall_bl_x]
wall_bl_z == Divide(2;f0.1;f0.8;f0.1)[restWall;lastWindow;rest
wall_bl_x == Divide(0;f0.1;f0.8;f0.1)[restWall;lastWindow;rest
lastWindow == GeometryDraw(plain)[windows]
ground == PlaneS(sides)[frontwall;wallground;wallsides;wallsi
frontwall == GeometryDraw(plain)[metal]
wallsides == GeometryDraw(plain)[metal]
wallsides == GeometryDraw(plain)[metal]
wallground == GeometryDraw(plain)[metal]
roof == GeometryDraw(plain)[metal]
```



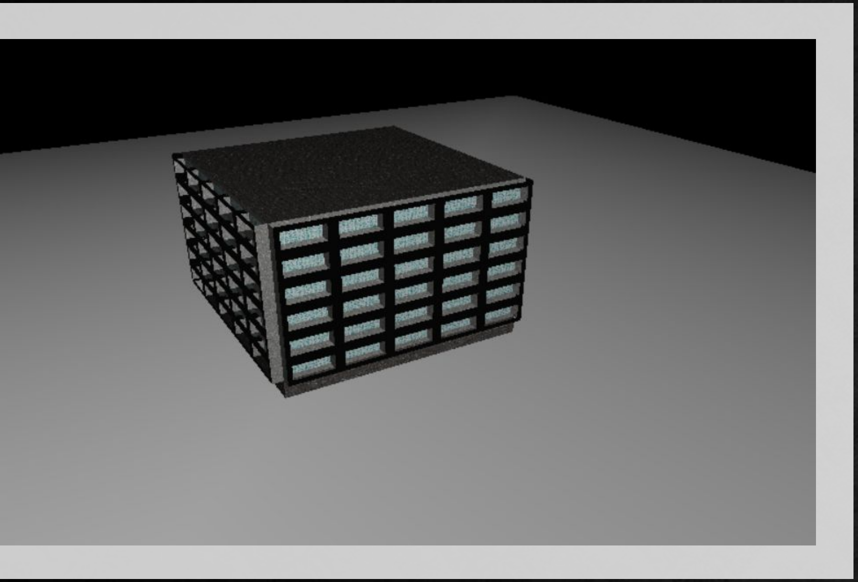
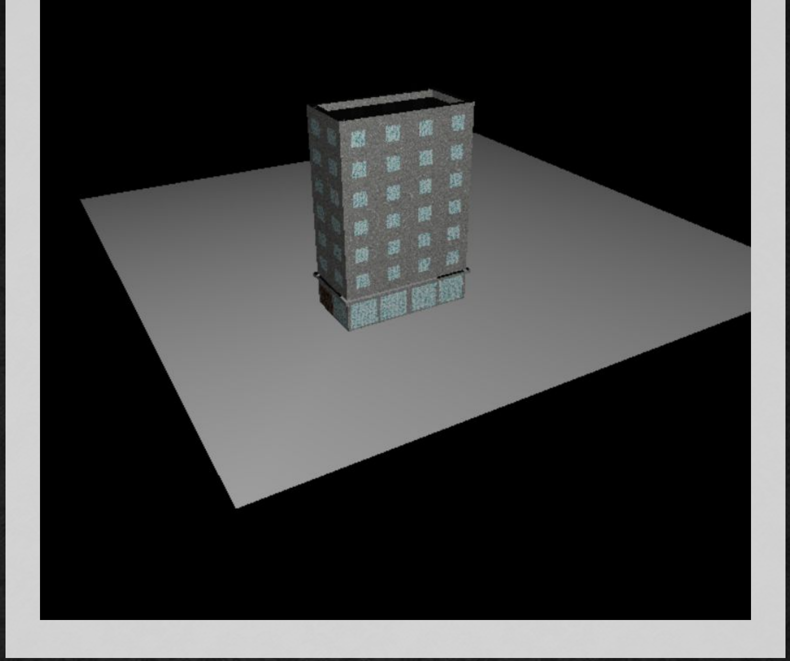
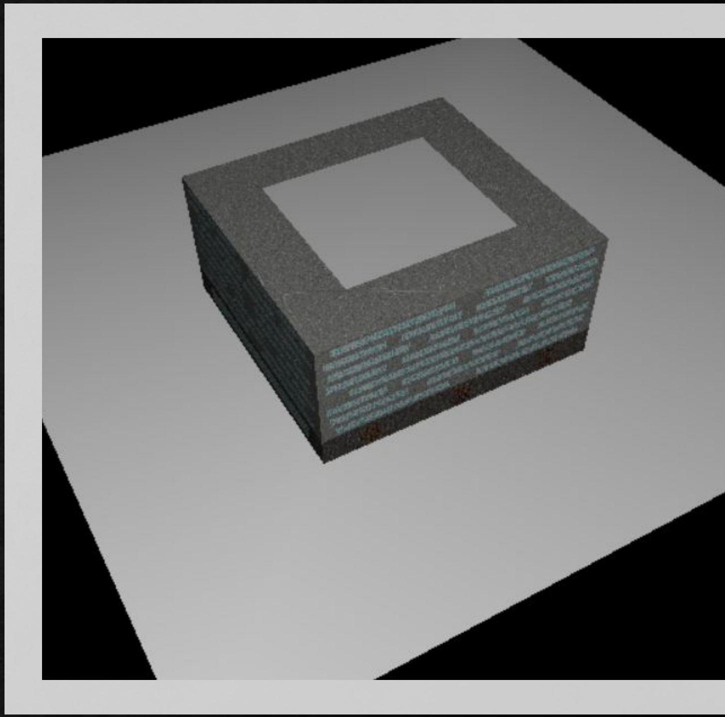
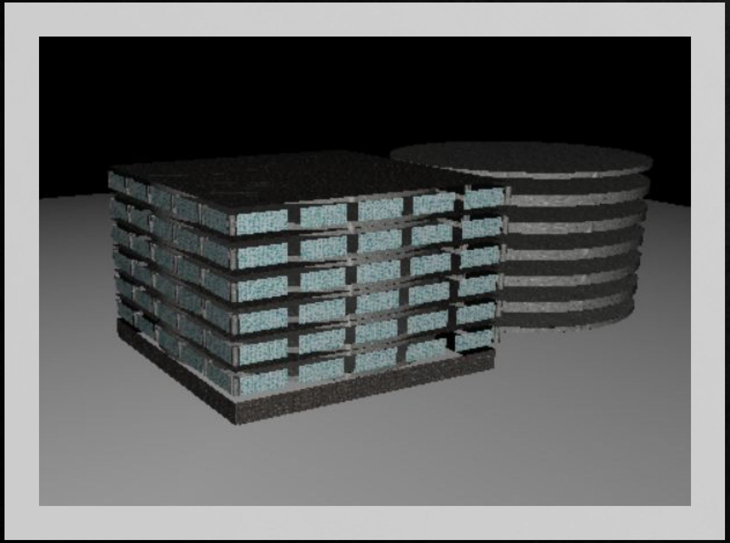
DEMO



```
                (*currentGeometry).clear();
                if (command.compare(shape.getName()) == 0) // apply rule if name in rule head matches
                    rule(shape);
                return (*currentGeometry); };
        }


};




//main function
int main(int argc, char** argv) {
    //Change bellow to run different worlds, all txt files should be within resource files
    // vvvvvvv change this  vvvvvv //
    Parser parser("basicbuildingRandom.txt");
    //Parser parser("basicbuilding.txt");
    //^^^^^^^^ change this ^^^^^^^  //

    //get rules given the file above
    auto rules = parser.parseRules();

    //make derivation tree, and use the rules to build it.
    derivationTree tree;
    tree.derivationTreeBuilder(rules, Geometry("main", Vec3(0, 0, 0), Vec3(40, 120, 80), WORLD));

    //get all the nodes from the dderivation tree
    auto nodes = tree.getLeafNodes();

    //draw world given the nodes goes to visualizer.cpp
    drawWorld(nodes, argc, argv);

    return 0;
}
```

Show output from: Debug

```
The thread 0x46e0 has exited with code 0 (0x0).
The thread 0x10cc has exited with code 0 (0x0).
The thread 0x2c30 has exited with code 0 (0x0).
The thread 0x528 has exited with code 0 (0x0).
The thread 0x5300 has exited with code 0 (0x0).
The thread 0x482c has exited with code 0 (0x0).
The thread 0x6d20 has exited with code 0 (0x0).
The program '[32088] BasicLabProject.exe' has exited with code 0 (0x0).
```

SOME MORE EXAMPLES

SOURCES

- Alkaim, A. (n.d.). *Procedural Generation for Architecture*. http://web.ist.utl.pt/antonio.menezes.leitao/Rosetta/FinalReport/reports/ArturAlkaim-Report.pdf.
- Muller, P., Wonka, P., Haegler, S., Ulmer, A., & Gool, L. V. (2004). *Procedural Modeling of Buildings*. http://peterwonka.net/Publications/pdfs/2006.SG.Mueller.ProceduralModelingOfBuildings.final.pdf.
- Nishida, G., Bousseau, A., & Aliaga, D. G. (2018, May 22). *Procedural Modeling of a Building from a Single Image*. Wiley Online Library. https://onlinelibrary.wiley.com/doi/10.1111/cgf.13372.
- *Procedural City Generator*. Henry Dai Gameplay Programmer. (n.d.). https://www.henrydai.net/procedural-city-generator.html.