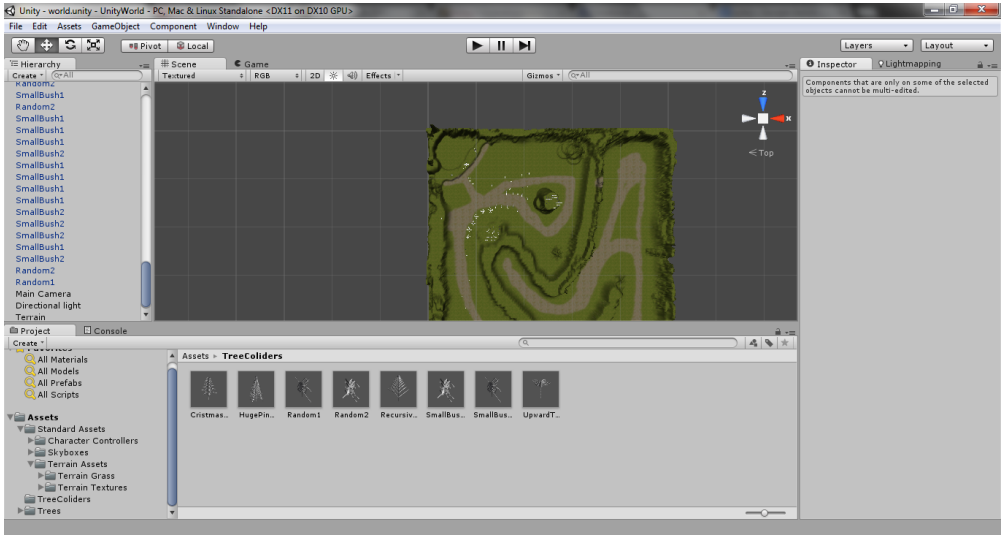
Procedural Modeling

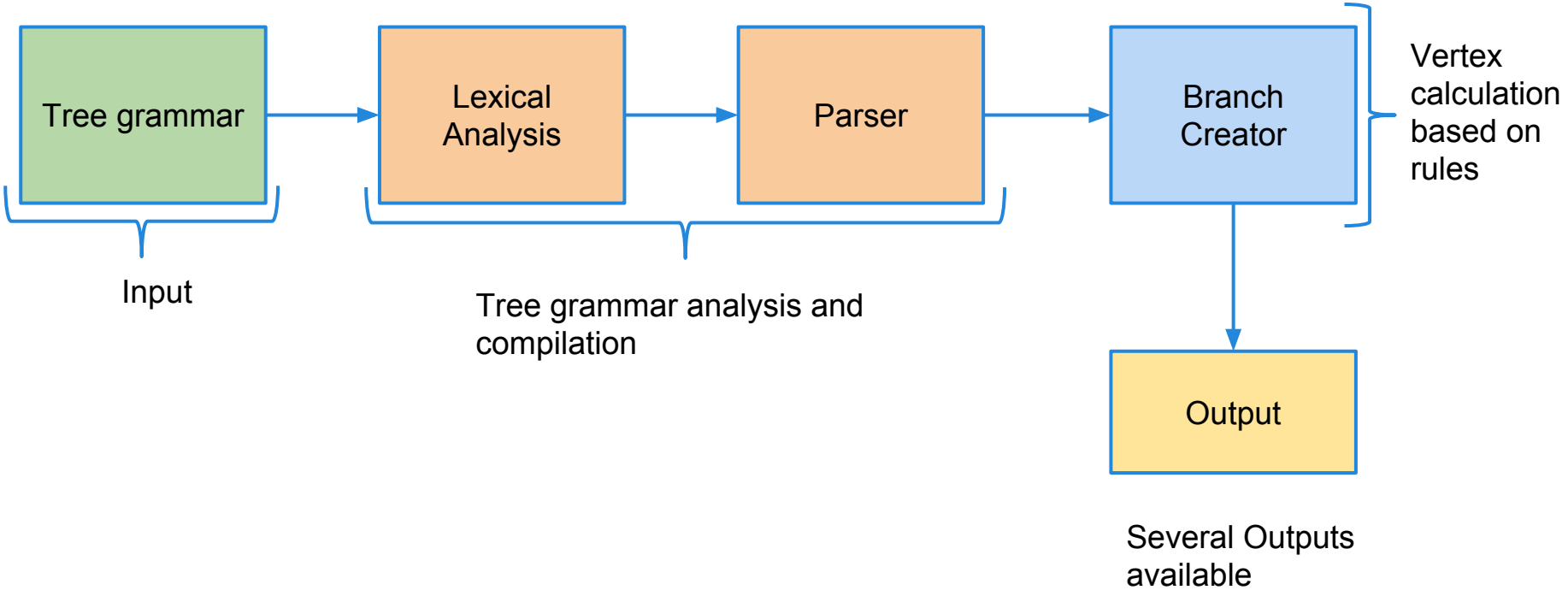
Santiago Velez Saffon

What my project can do!!

Show world in Unity



How does it work?? Project Pipeline



Lexical Analysis

```
digit       [0-9]
char        [a-zA-Z]
left        [+]
right       "-"
pitch_up    "^"
pitch_down  [&]
roll_left   [$]
roll_right  ":"
turn        "|"
semicol     [;]
assign      "="
left_bra    "["
right_bra   "]"
angle       "angle"
iter        "iter"
axiom       "axiom"
define      "#define"
render      "render()"
name        "name"
variables   "variables"
move        "move"
direction   "direction"
classname   "class"
length      "length"
```

Characters and numbers

Operators

Reserved words

1. Understand The input
2. Extract important values.
3. Remove unwanted characters.

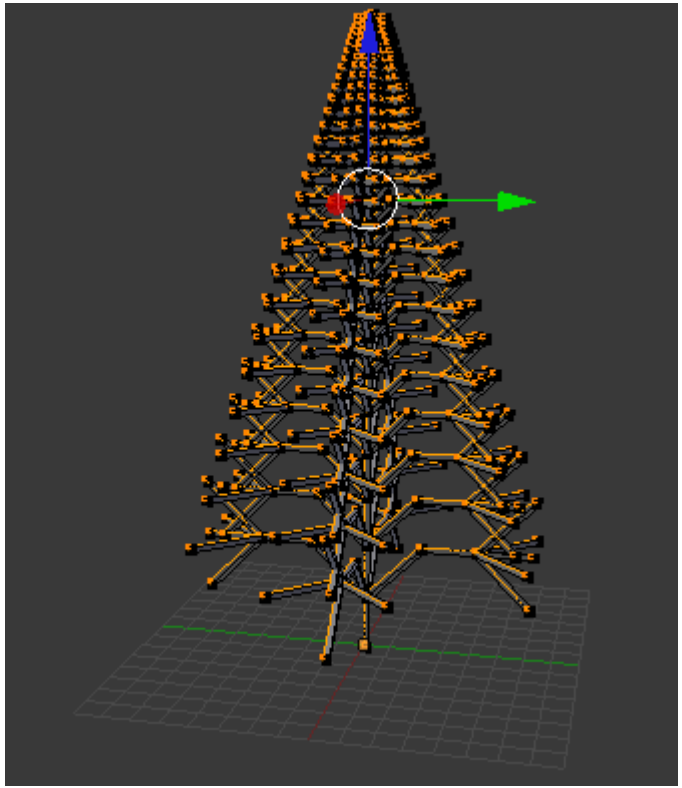
Parsing

```
l_system    :
            list_definitions axiom equations RENDER SEMICOL {render();}
            |axiom equations RENDER SEMICOL {render();}

axiom       :
            AXIOM EQUAL word SEMICOL    {isaxiom=0;axiom();}

equations   :
            equations equation
            |equation

equation    :
            VAR_NAME EQUAL expression SEMICOL   {int res; res=equation($1); if(!res){yyerror("Undefined constant in rule");};}

expression  :
            expression word
            |expression operator
            |operator
            |word

word        :
            word VAR_NAME           {(isaxiom==1) ? add_to_word($2):add_operator($2);}
            |VAR_NAME               {(isaxiom==1) ? add_to_word($1):add_operator($1);}

operator    :
            LEFT                    {add_operator($1);}
            |RIGHT                  {add_operator($1);}
            |ROT_Z_POS              {add_operator($1);}
            |ROT_Z_NEG              {add_operator($1);}
            |ROT_Y_POS              {add_operator($1);}
            |ROT_Y_NEG              {add_operator($1);}
            |ROT_X                  {add_operator($1);}
            |LBRA                   {add_operator($1);}
            |RBRA                   {add_operator($1);}
```

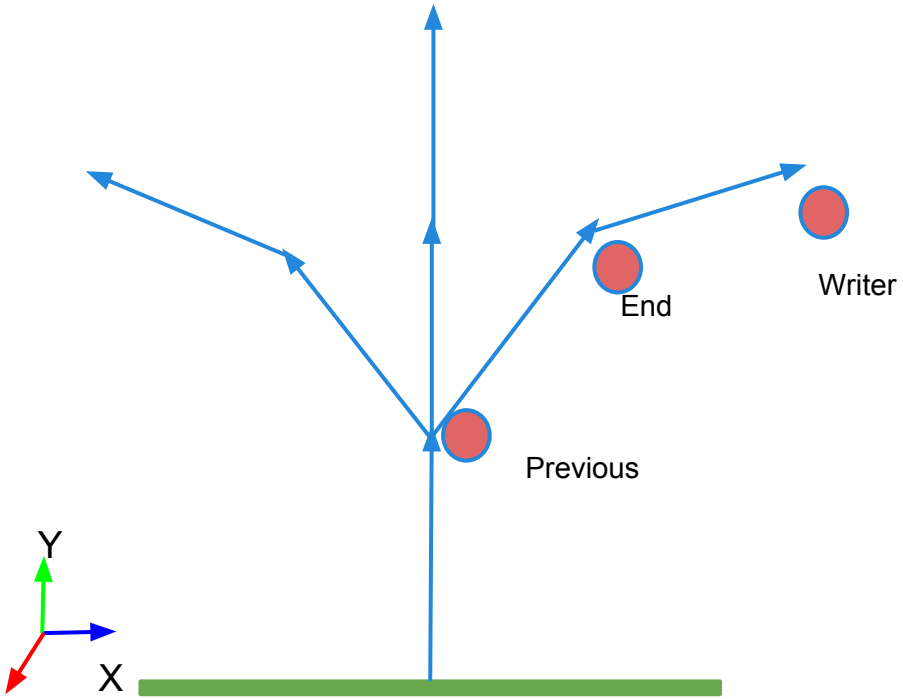
1. Build relevant information from user's input.
2. Iterations, angle, rules axiom,

Grammar Example

```
#define angle=45;
#define iter=15;
#define length=80;
#define name=smalltree;
#define class=SmallTree;
#define variables=FXBT;
#define move=FBT;
axiom=X;
F=[-T[:T][|T]T];
B=[+T[&T][$T]T];
X=T[F][B]X;
T=T;
render();
```



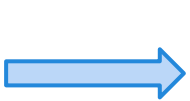
Branch Creation



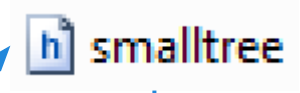
1. 3 sentinels to help build the tree.
   a. Previous
   b. End
   c. Writer.
2. Tree grows towards positive Y axis.
3. All rotation are first calculated on writer. Direction is calculated and finally end is created.
4. End - previous form a branch.

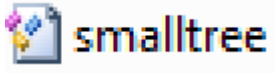
Output

```
#define angle=45;
#define iter=15;
#define length=80;
#define name=smalltree;
#define class=SmallTree;
#define variables=FXBT;
#define move=FBT;
axiom=X;
F=[-T[:T][|T]T];
B=[+T[&T][$T]T];
X=T[F][B]X;
T=T;
render();
```



project.exe

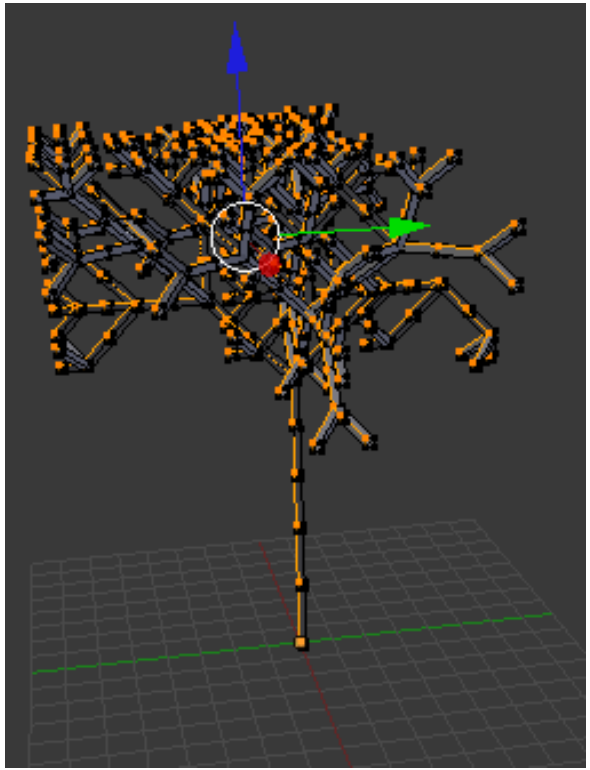


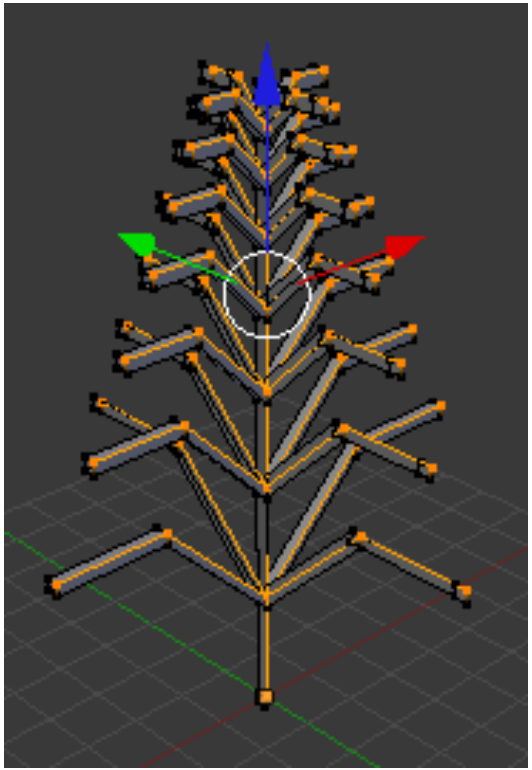


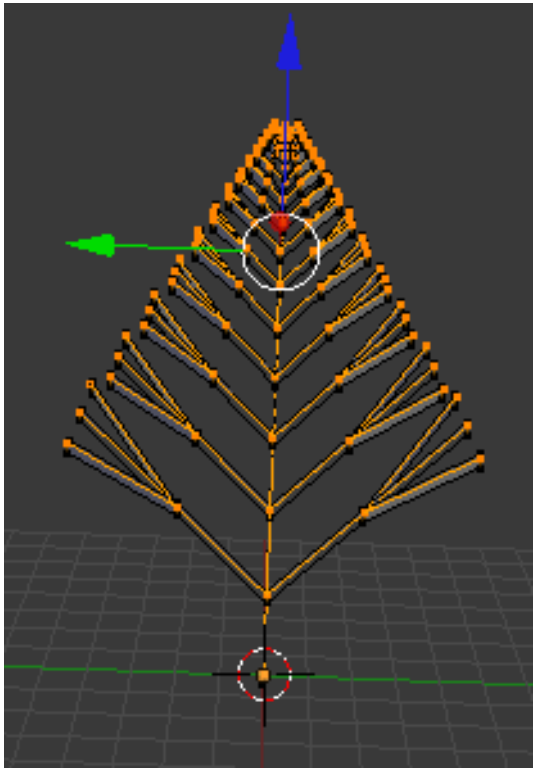
C++ Header file, that contains all the properties and algorithms to build a tree.

ASCII wavefront .obj. Which represents the tree connectivity.

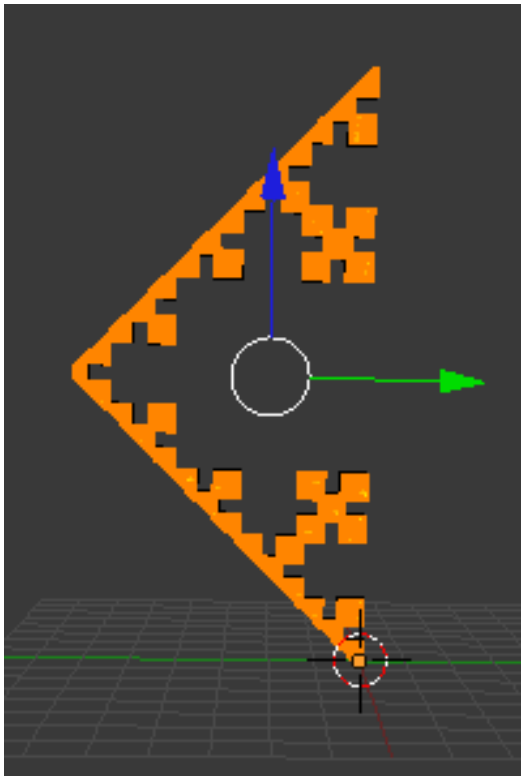
Tree examples

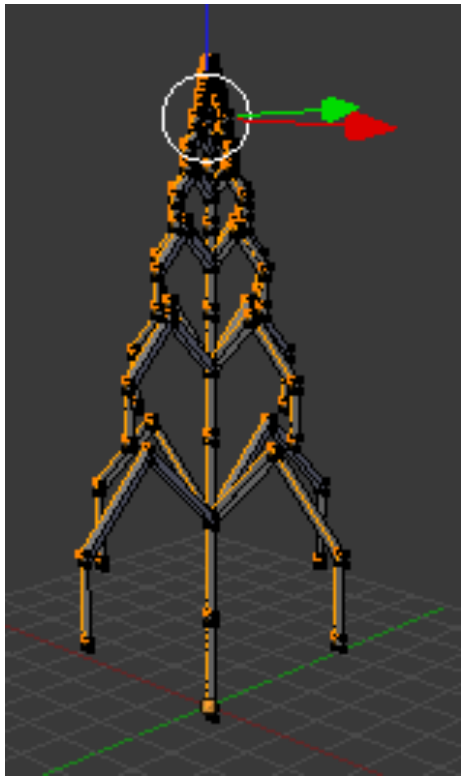


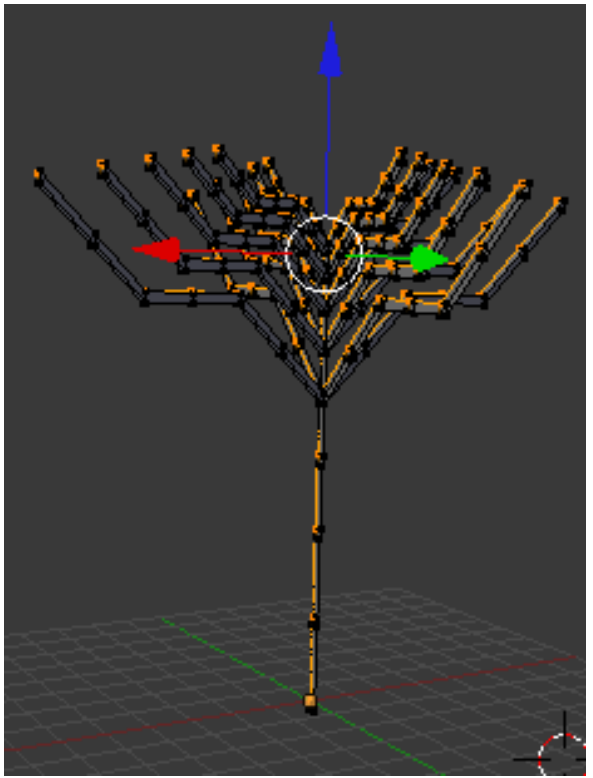




Tree examples

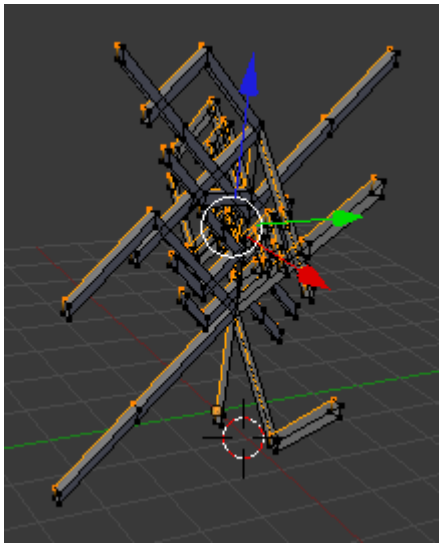


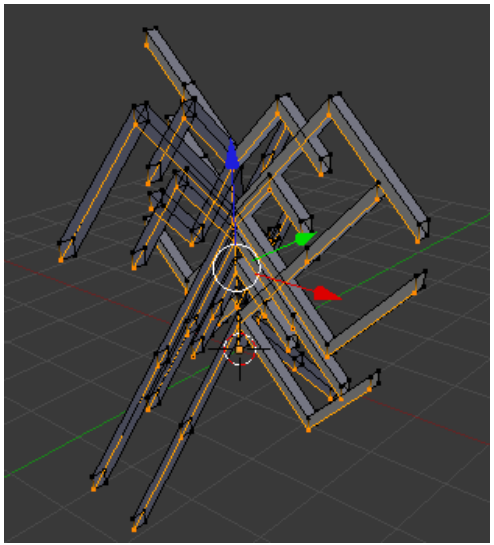




Randomness??

1. Yes, but not so good for big trees





Difficulties- Problems

1. Some Rotation can be visually useless.

Tree growing towards Y axis; rotation in Y is useless.

2. Very easy to get lost on the tree.
   a. With a big number of iterations its very easy to get a lot of useless rotations.

3. Making real trees require much more complex grammars.randomness??

4. Real trees and not so recursive. On a real tree every branch is different.

Future work

1. Extend grammar to allow several angles for a single tree.
   a. At least angle per type of rotation.

2. Extend grammar to allow leaf creation.

3. A better management of randomness.
4.

Resources

1. Flex -http://flex.sourceforge.net/
2. Byson- http://gnuwin32.sourceforge.net/packages/bison.htm
3. Unity-http://unity3d.com/
4. Blender-http://www.blender.org/
5. The Algorithmic Beauty of Plants - http://algorithmicbotany.org/papers/#abop