BLOX

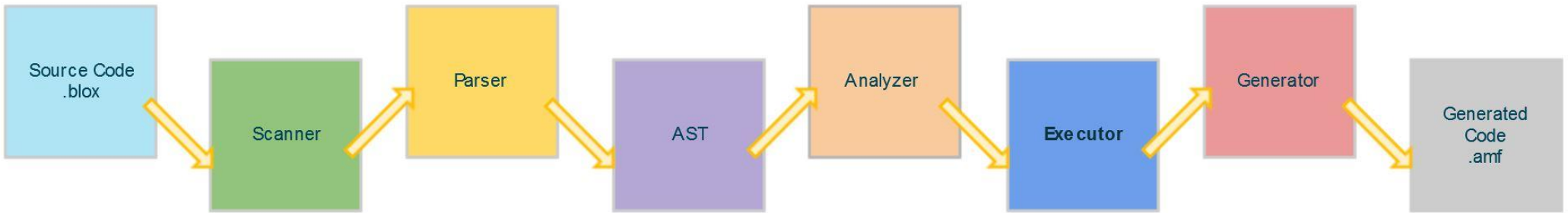
Naeem Bhatti
Jonathan Voss
Tyrone Wilkinson

Motivation

- Computer-aided design and computer-aided engineering software such as Solidworks allow the user to model 2D and 3D structures by providing a series of parameters. Sandbox video games such as Minecraft enable users to build 3D structures out of cubes. Neither of these program allow the user to solve a structural program. This is where Blox comes in.

- Given a set of resources and a mapping of constraints on those resources, Blox can be used to output the set of all possible solutions that conform to those mappings in the Additive Manufacturing File format (AMF). The generated .amf file is 3D printable, allowing for further enhancements in any compatible modeling program, or it can be directly printed in any compatible printer.

Architecture



- Scanner: Character Stream → Token Stream
- Parser: Token Stream → Abstract Syntax Tree (AST)
- Analyzer: AST → A Semantically Checked AST
- Executor: Contains the underlying code for:
  - The functions our language provides for the programmer's use
  - Checking the validity of the programmer's use
- Generator: Generates an .amf file based on the frame the programmer wants to be printed

Language Tutorial

- Blox is very beginner-friendly with a syntax similar to C and Java.
- The programmer can make use of primitive data types (int, float, string, bool), aggregate data types (array), and loops (for, while) for a smooth programming experience.
- Programmer-defined functions are an unrequired but optional aspect of our language.
- Like in C, the “main” function is the designated entry point of the program and is called at program startup.

Language Example 1

```
1  int main()
2  {
3    /* Print string, int, float, bool */
4    print("hello");
5    print(10);
6    print(10.5);
7    print(true);
8    print(false);
9  }
```

Our inbuilt print function automatically detects the primitive type, which it then streams to standard output.

Language Example 2

```
1  /* Global frame declarations */
2  Frame<2,1,1> twox;
3  Frame<1,5,1> fivey;
4  Frame<46,5,1> base;
5
6  /* Global frame assignment */
7  Frame H = fivey;
8
9  int main()
10 {
11   /* Local face declarations */
12   Face<1,3,1,E> f1;
13   Face<1,3,1,W> f2;
14   Face<1,1,1,W> f3;
15   Face<3,3,1,E> f4;
16   Face<1,1,1,F> f5;
17   Face<1,1,1,B> f6;
18
19   /* Join calls */
20   Join(H, f1, twox, f3);
21   Join(H, f4, fivey, f2);
22   Join(base, f5, H, f6);
23
24   /* Convert call */
25   Convert(base);
26 }
```

- Let's call this short program "H." As is, the generated .amf file would produce the letter 'H' on a base with the dimensions 46x5x1.

- The .amf file and model can be found in the following slides.
  - Spaces and comments have been added to the .amf file for improved readability.
  - An actual image of the model on actual 3D printing software.

Language Example 2 .amf

```
<?xml version="1.0" encoding="utf-8"?>
<amf>
  <object id="1">
    <mesh>
      <vertices>
        <!-- 0 -->
        <vertex><coordinates><x>46</x><y>5</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>46</x><y>5</y><z>0</z></coordinates></vertex>
        <vertex><coordinates><x>46</x><y>0</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>46</x><y>0</y><z>0</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>5</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>5</y><z>0</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>0</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>0</y><z>0</z></coordinates></vertex>

        <!-- 8 -->
        <vertex><coordinates><x>1</x><y>5</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>1</x><y>5</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>1</x><y>0</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>1</x><y>0</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>5</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>5</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>0</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>0</x><y>0</y><z>1</z></coordinates></vertex>

        <!-- 16 -->
        <vertex><coordinates><x>3</x><y>3</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>3</x><y>3</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>3</x><y>2</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>3</x><y>2</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>1</x><y>3</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>1</x><y>3</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>1</x><y>2</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>1</x><y>2</y><z>1</z></coordinates></vertex>

        <!-- 24 -->
        <vertex><coordinates><x>4</x><y>5</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>4</x><y>5</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>4</x><y>0</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>4</x><y>0</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>3</x><y>5</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>3</x><y>5</y><z>1</z></coordinates></vertex>
        <vertex><coordinates><x>3</x><y>0</y><z>2</z></coordinates></vertex>
        <vertex><coordinates><x>3</x><y>0</y><z>1</z></coordinates></vertex>
      </vertices>
```

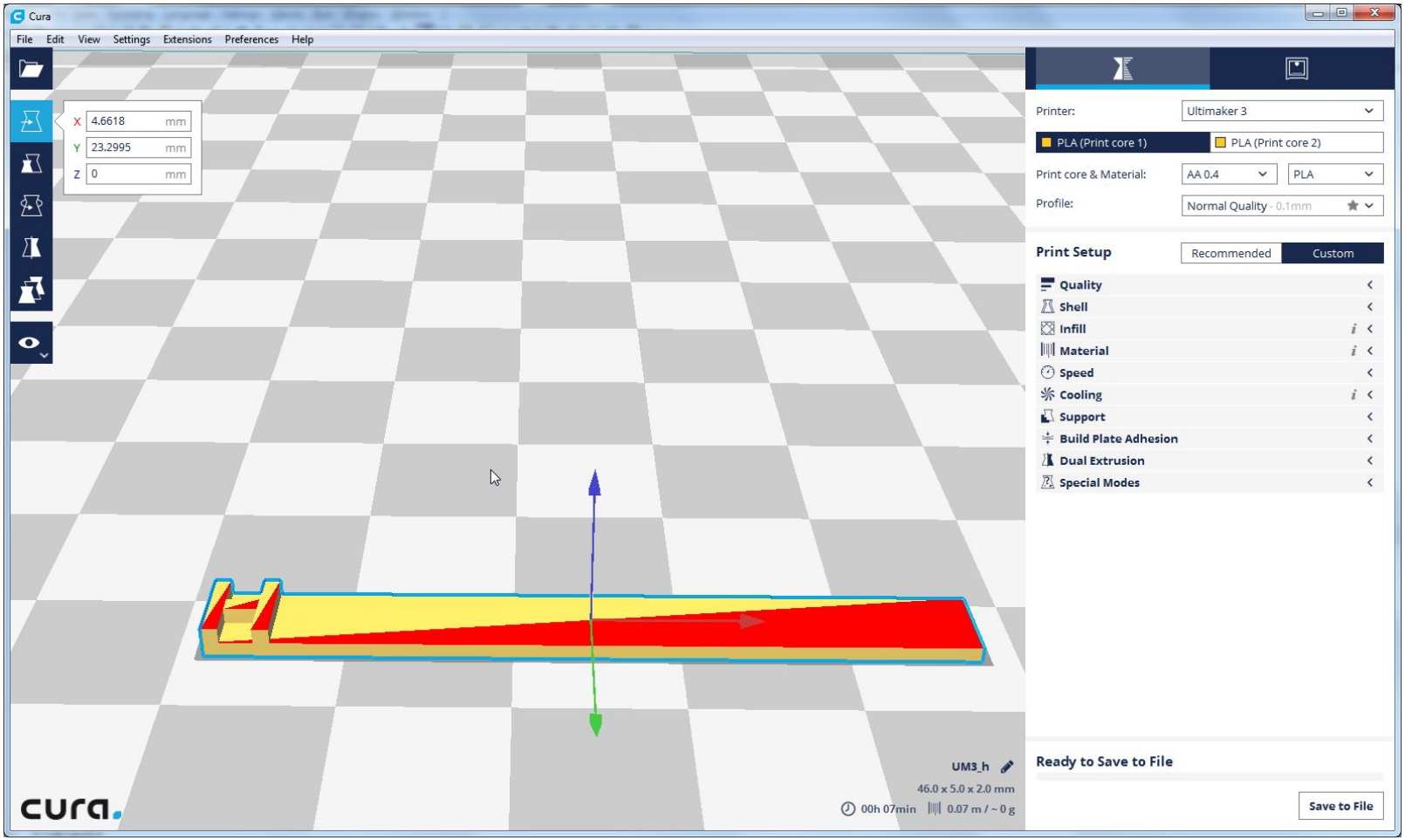
Language Example 2 .amf

```
<volume>
  <!-- Right Face -->
  <triangle><v1>0</v1><v2>1</v2><v3>3</v3></triangle>
  <triangle><v1>0</v1><v2>2</v2><v3>3</v3></triangle>
  <!-- Left Face -->
  <triangle><v1>5</v1><v2>6</v2><v3>7</v3></triangle>
  <triangle><v1>4</v1><v2>5</v2><v3>7</v3></triangle>
  <!-- Top Face -->
  <triangle><v1>0</v1><v2>1</v2><v3>5</v3></triangle>
  <triangle><v1>0</v1><v2>4</v2><v3>5</v3></triangle>
  <!-- Bottom Face -->
  <triangle><v1>2</v1><v2>3</v2><v3>7</v3></triangle>
  <triangle><v1>2</v1><v2>6</v2><v3>7</v3></triangle>
  <!-- Front Face -->
  <triangle><v1>0</v1><v2>2</v2><v3>6</v3></triangle>
  <triangle><v1>0</v1><v2>4</v2><v3>6</v3></triangle>
  <!-- Back Face -->
  <triangle><v1>1</v1><v2>3</v2><v3>7</v3></triangle>
  <triangle><v1>1</v1><v2>5</v2><v3>7</v3></triangle>

  <!-- "H" -->
  <!-- Right Face -->
  <triangle><v1>8</v1><v2>9</v2><v3>11</v3></triangle>
  <triangle><v1>8</v1><v2>10</v2><v3>11</v3></triangle>
  <!-- Left Face -->
  <triangle><v1>13</v1><v2>14</v2><v3>15</v3></triangle>
  <triangle><v1>12</v1><v2>13</v2><v3>15</v3></triangle>
  <!-- Top Face -->
  <triangle><v1>8</v1><v2>9</v2><v3>13</v3></triangle>
  <triangle><v1>8</v1><v2>12</v2><v3>13</v3></triangle>
  <!-- Bottom Face -->
  <triangle><v1>10</v1><v2>11</v2><v3>15</v3></triangle>
  <triangle><v1>10</v1><v2>14</v2><v3>15</v3></triangle>
  <!-- Front Face -->
  <triangle><v1>8</v1><v2>10</v2><v3>14</v3></triangle>
  <triangle><v1>8</v1><v2>12</v2><v3>14</v3></triangle>
  <!-- Back Face -->
  <triangle><v1>9</v1><v2>11</v2><v3>15</v3></triangle>
  <triangle><v1>9</v1><v2>13</v2><v3>15</v3></triangle>

  <!-- Right Face -->
  <triangle><v1>16</v1><v2>17</v2><v3>19</v3></triangle>
  <triangle><v1>16</v1><v2>18</v2><v3>19</v3></triangle>
  <!-- Left Face -->
  <triangle><v1>21</v1><v2>22</v2><v3>23</v3></triangle>
```

Language Example 2 .amf

```
91          <triangle><v1>20</v1><v2>21</v2><v3>23</v3></triangle>
92          <!-- Top Face -->
93          <triangle><v1>16</v1><v2>17</v2><v3>21</v3></triangle>
94          <triangle><v1>16</v1><v2>20</v2><v3>21</v3></triangle>
95          <!-- Bottom Face -->
96          <triangle><v1>18</v1><v2>19</v2><v3>23</v3></triangle>
97          <triangle><v1>18</v1><v2>22</v2><v3>23</v3></triangle>
98          <!-- Front Face -->
99          <triangle><v1>16</v1><v2>18</v2><v3>22</v3></triangle>
100         <triangle><v1>16</v1><v2>20</v2><v3>22</v3></triangle>
101         <!-- Back Face -->
102         <triangle><v1>17</v1><v2>19</v2><v3>23</v3></triangle>
103         <triangle><v1>17</v1><v2>21</v2><v3>23</v3></triangle>
104
105         <!-- Right Face -->
106         <triangle><v1>24</v1><v2>25</v2><v3>27</v3></triangle>
107         <triangle><v1>24</v1><v2>26</v2><v3>27</v3></triangle>
108         <!-- Left Face -->
109         <triangle><v1>29</v1><v2>30</v2><v3>31</v3></triangle>
110         <triangle><v1>28</v1><v2>29</v2><v3>31</v3></triangle>
111         <!-- Top Face -->
112         <triangle><v1>24</v1><v2>25</v2><v3>29</v3></triangle>
113         <triangle><v1>24</v1><v2>28</v2><v3>29</v3></triangle>
114         <!-- Bottom Face -->
115         <triangle><v1>26</v1><v2>27</v2><v3>31</v3></triangle>
116         <triangle><v1>26</v1><v2>30</v2><v3>31</v3></triangle>
117         <!-- Front Face -->
118         <triangle><v1>24</v1><v2>26</v2><v3>30</v3></triangle>
119         <triangle><v1>24</v1><v2>28</v2><v3>30</v3></triangle>
120         <!-- Back Face -->
121         <triangle><v1>25</v1><v2>27</v2><v3>31</v3></triangle>
122         <triangle><v1>25</v1><v2>29</v2><v3>31</v3></triangle>
123       </volume>
124     </mesh>
125   </object>
126 </amf>
```

Language Example 2 model



Manipulate to your liking then print when ready!

Language Examples

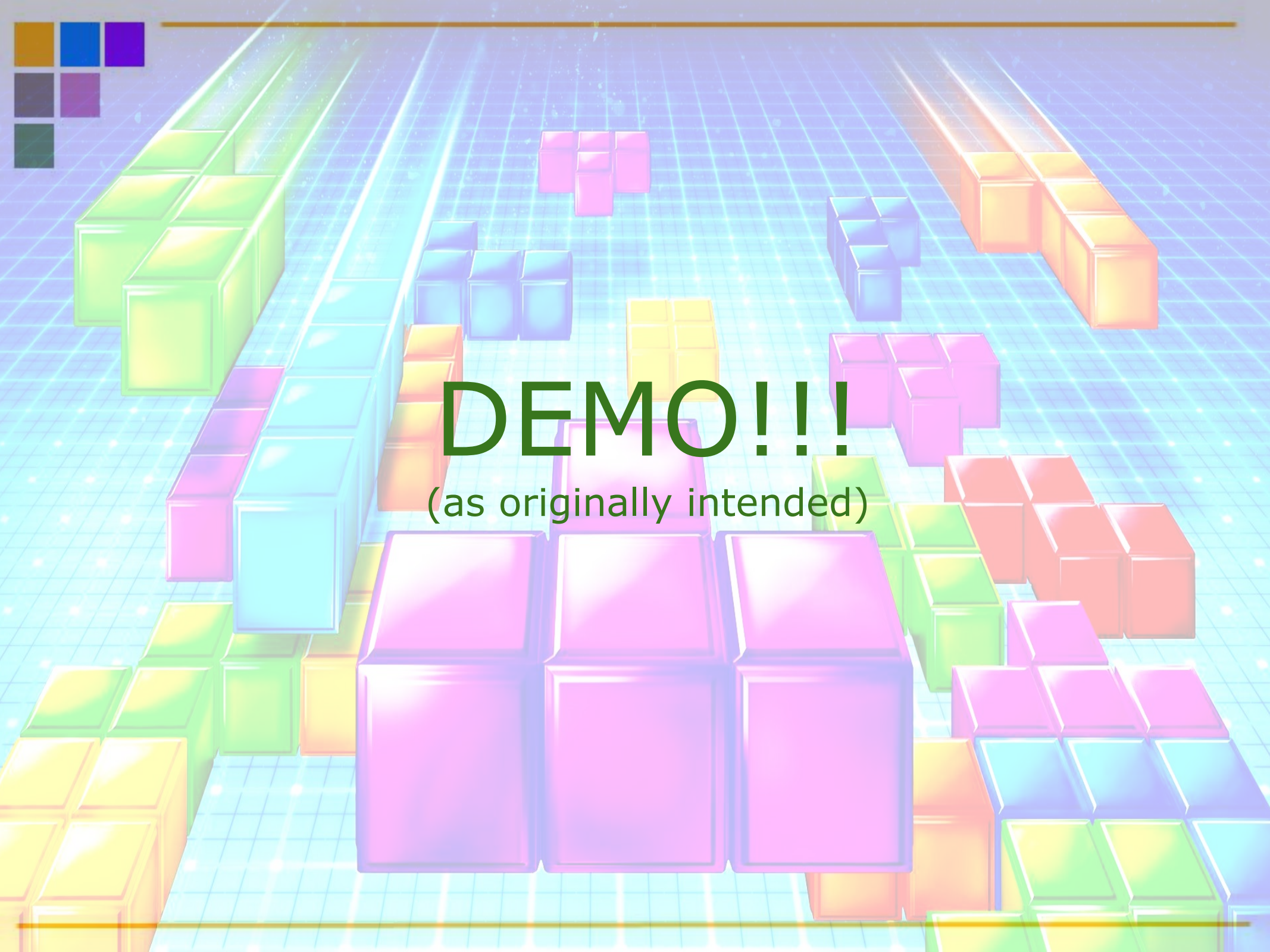
The prior examples are just a small, elementary sample of what our language has to offer. You can be as simple or elaborate as you'd like.

Important Lessons Learned

- Scheduling
- Time Management
- Effective Communication
- Consistent Collaboration
- Laying out the Language (seeing the big picture) early
- Understanding compiler components early
- OCAML

Possible Language Expansion

- Time and resource constraints forced us to reduce the capabilities of our language. But it was designed to be flexible enough for features to be added later on. Some of these additions could include:
  - Expanding the problem solving ability of Blox. This could take the form of allowing the programmer to design more complex rules upon which object generation would be regulated by.
  - Increasing the robustness of our compiler so that users could specify their intentions in a high level way so that they would not have to think in terms of blocks. Perhaps our solution resides in machine learning.
  - Incorporating features present in the AMF open standard such as material and texture definition to give programmers greater control over their creations.


DEMO!!!
(as originally intended)

Questions?
Comments?