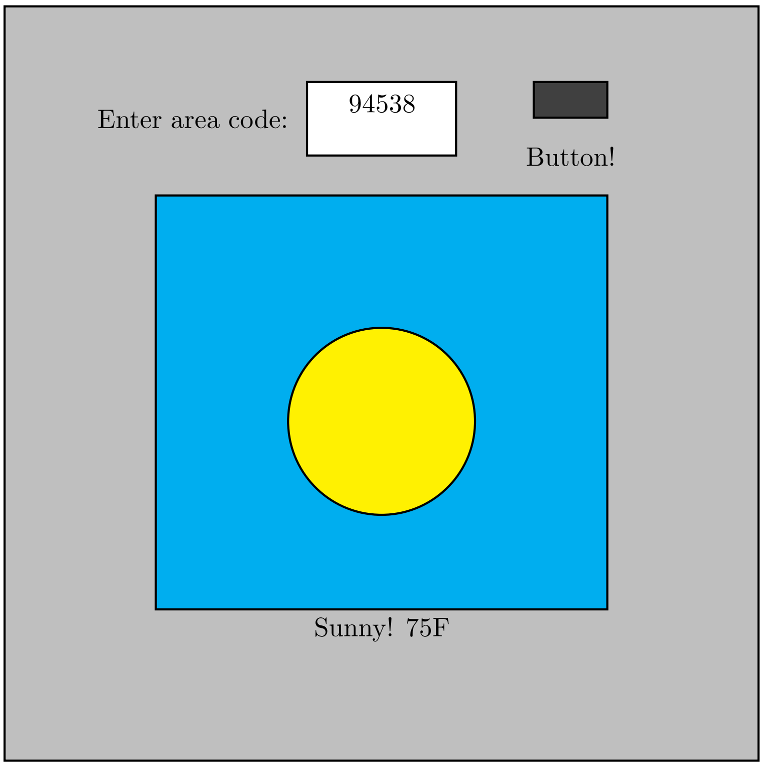
Daniel Hulseman Ted Wang Sahil Chopra Weihan Zhang

1 Specification

We will be making an app that will prompt the user for either/or a zip code or town name, and return the weather for that area. The weather for the area will be represented by a picture.

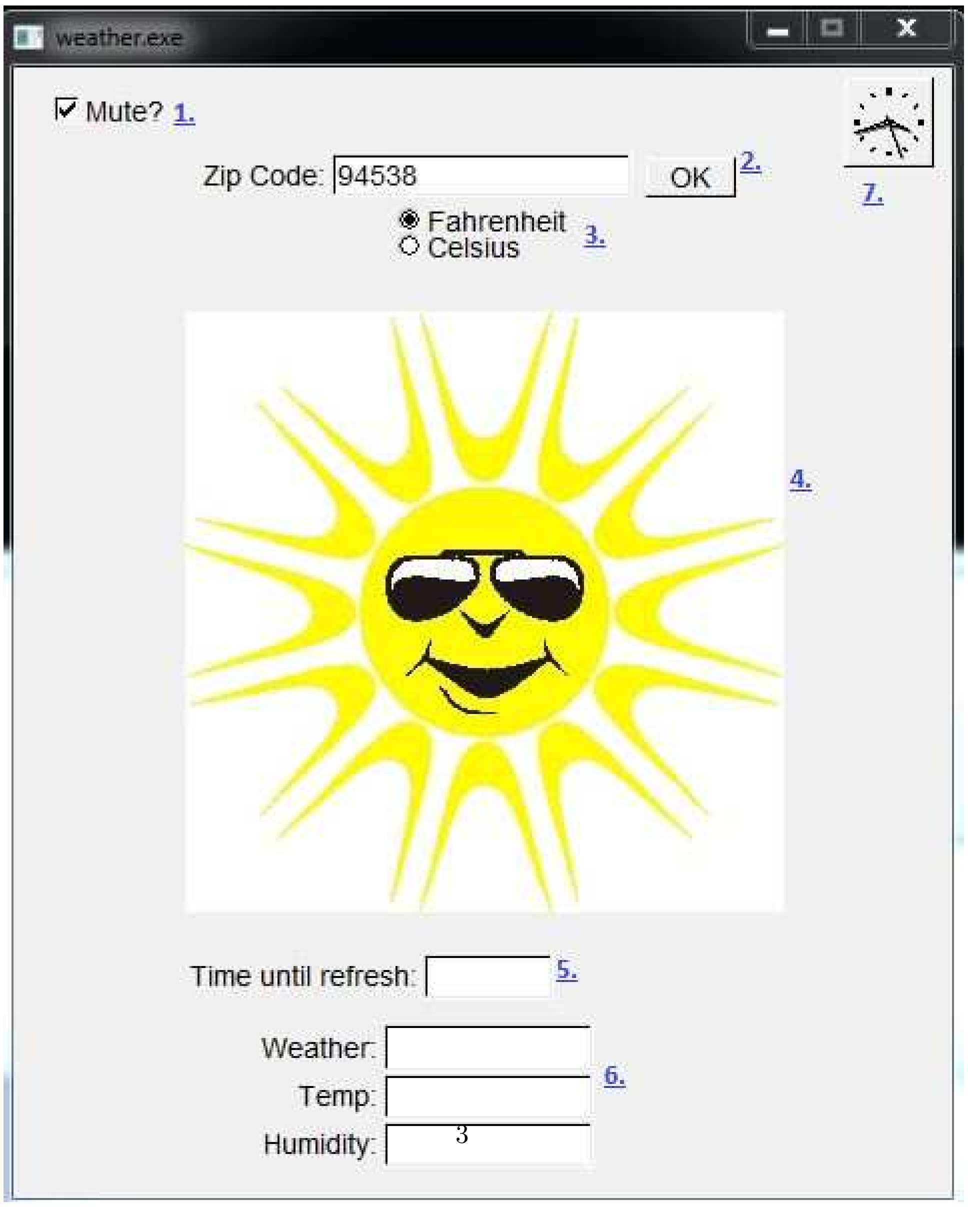
Goals:
- User will be able to choose which area he/she wants to check the weather
- 1-7 day forecast will be returned along with other information, such as a preference of Celsius or Fahrenheit
- Pictures and sound appropriate for the weather

Things we'd like to do:
- Save what the weather was last time you ran the program
- Custom animation for the weather using FLTK
- Choose which day you want to see the weather for
- Mute button!



2 Analysis

IPO: Inputs are the zip code from the user, and the current weather taken from the internet, in addition to a preference of Fahrenheit or Celsius. Output is the weather, temperature, humidity, time, etc, represented by pictures/animation and sound. Process could entail a timer from the last time you requested weather, and conversions from Fahrenheit to Celsius or vice versa and checking to get the appropriate picture/sound for the weather.
Example:
-User enters 94538 as the zip code, and sets Fahrenheit
-Program displays a sun, while birds chirp and peaceful music plays. Displays 77F.
-Program begins to count how long ago he requested the weather.
Example 2 (invalid zip code):
-User enters 1234 as the zip code.
-Program displays a red X where the weather would have been and displays "Invalid zip code" where the temperature would have been.



[✓] Mute?

Zip Code: 94538

[● Fahrenheit]
○ Celsius

Time until refresh:

Weather:

Temp:

Humidity:

1. Mute function: Input is the check box and whether or not it is checked. Output is either muted sound or allowed sound. We aren't sure how to do this yet so help is appreciated.

2. This is the meat of the program. The user must input a zip code and hit the OK button when ready. Output is the weather represented symbolically in function 4 and details in function 6. Process retrieving the data from a database.

3. Temperature preference: Input is done via the radial buttons picking the preferred unit of temperature. Output is displayed in function 6 giving you your preferred units. Process is done by calculating fahrenheit from celsius or vice versa depending on our database's units.

4. Image selection: Input is acquired by function 2's output. Output is the appropriate image according to the weather. The process is similar to lab13: there are actually multiple images overlayed on top of each other. In order to display the appropriate weather the program must display the image with the appropriate value and hide all the others that don't apply.

5. Refresh Timer: Input once again comes from function 2's OK button. Output is simply time counting down from when that button was pressed until data is re-retrieved. This process is done by using the FLTK(?) timeout function counting down from when this button was pressed, and then returning data for the same zip code again.

6. Weather details: Input is acquired from data from the internet acquired from function 2 and also temperature determined by function 3.Output is details about weather conditions from this database. Process is to acquire data from the internet.

7. Time (relative to the zipcode): Function 2 provides the input for this function. Output is time relative to the area that we specified. For example, if the area code is for Fremont, it display Pacific standard time. Process is to retrieve data from the internet relevant to this.

3 Design

We wrote a cpp file that retrieves data, finds data, and converts data from internet first so we'll know what and how to do inside FLuid. Then used FLuid to create the cxx source file. Here is what is produced:

4 Implementation

```
// generated by Fast Light User Interface Designer (fluid) version 1.0108

#include "weather.h"
#include<iostream>
#include<iomanip>
#include<sstream>
#include<cstring>
#include <cstdlib>
#include "curl.h"
#include "data_retrieve.h"
using namespace std;

double f2c(double f) {
  const double K = 0.55;
int c = (f - 32) * K;
return c;
}

Fl_Double_Window *Window=(Fl_Double_Window *)0;

Fl_Input *input2=(Fl_Input *)0;

Fl_Button *calculate=(Fl_Button *)0;

static void cb_calculate(Fl_Button*, void*) {
  const string URL = "http://www.wunderground.com/cgi-bin/findweather/getFore
istringstream iss(input2->value());
string zipcode;
string website = URL+iss.str();
cout<<website;
string page;
ostringstream oss;
CURL* curl = curl_easy_init(); // a "handle" or "pointer" into the library
if(curl) // check to make sure if the handle is valid
{
        curl_easy_setopt(curl,CURLOPT_WRITEFUNCTION,handle_data);
        curl_easy_setopt(curl,CURLOPT_WRITEDATA,&page);
        curl_easy_setopt(curl,CURLOPT_URL,website.c_str());
        CURLcode result = curl_easy_perform(curl);
        if(result==0)
        {
                cout<<"access successful"<<endl;
        }
        else
        {
                cout<<"Access failed. Please try again later."<<endl;
                curl_easy_cleanup(curl);
        }
}
else
{
cout<<"Error in curl handle"<<endl;
exit(1); // failure to get exchange rate, so quit program
}

        // if user checks Farenheit, display temperature unit in Farenheit
if( farenheit -> value() )
{
// Finds the current temperature
size_t pos_of_clear = page.find("+Clear+");
size_t tempnow_ld = page.find("(", pos_of_clear);
size_t tempnow_rd = page.find("+", tempnow_ld);
string tempnow = page.substr(tempnow_ld + 1 , tempnow_rd - tempnow_ld - 1);

        if( tempnow.length() > 5) // quits the program if it doesn't find +Cl
        {
                cout<<"Error: Wrong zip code "<<endl;
                cout<<"Please enter a valid zip code: ";
                //input = 0;
                //cin>>input;
                //ss.clear();
                //ss.str("");
                //ss << input;
        }
cout<<"The current temperature is "<<tempnow<<" F";
oss << tempnow;
temperature -> value(oss.str().c_str());
oss.clear();
oss.str("");

//  This is to find High of the day
size_t pos_of_later = page.find("Later%3A+");
size_t high_ld = page.find("+("  , pos_of_later);
size_t high_rd = page.find("+F"  , high_ld);
string high = page.substr(high_ld + 2 , high_rd - high_ld - 2);
oss << high;
cout<<"Today's High is "<<high<<" F"<<endl;
hight -> value(oss.str().c_str());
oss.clear();
oss.str("");

//  This is to find Low of the day
size_t low_ld = page.find("%2F+" , pos_of_later);
size_t low_rd = page.find("F)" , low_ld);
string low = page.substr( low_ld + 4 , low_rd - low_ld - 5);
oss << low;
cout<<"Today's Low is "<<low<<" F"<<endl;
lowt -> value(oss.str().c_str());
oss.clear();
oss.str("");

// This finds the current humidity level
size_t pos_of_pwsrtb = page.find("pwsrt_b");
size_t h_ld = page.find("value=" , pos_of_pwsrtb);
size_t h_rd = page.find("><" , h_ld);
string humidity_now = page.substr( h_ld + 7 , h_rd - h_ld - 8 ) + " %";
oss << humidity_now;
cout<<"Today's Humidity Level is "<<humidity_now<<" %"<<endl;
humidity -> value(oss.str().c_str());
oss.clear();
oss.str("");


// This finds the current weather
size_t curw_ld = page.find("%3A+");
size_t curw_rd = page.find("+(" , curw_ld);
string curw = page.substr( curw_ld + 4  , curw_rd - curw_ld - 4 );
oss << curw;
cout<<"The current weather is "<<curw<<endl;
weather -> value(oss.str().c_str());
oss.clear();
oss.str("");
}
else if( celsius -> value() )
{
stringstream ss;

// Finds the current temperature
size_t pos_of_clear = page.find("+Clear+");
size_t tempnow_ld = page.find("(", pos_of_clear);
size_t tempnow_rd = page.find("+", tempnow_ld);
string tempnow = page.substr(tempnow_ld + 1 , tempnow_rd - tempnow_ld - 1);

        if( tempnow.length() > 5) // quits the program if it doesn't find +Cl
        {
```

```
                        cout<<"Error:␣Wrong␣zip␣code␣"<<endl;
                        cout<<"Please␣enter␣a␣valid␣zip␣code:␣";
                        //input = 0;
                        //cin>>input;
                        //ss.clear();
                        //ss.str("");
                        //ss << input;
                }
        double tnow = atof(tempnow.c_str());
        cout<<"Currently␣"<<f2c(tnow)<<"␣C";
        ss << f2c(tnow);
        temperature -> value(ss.str().c_str());
        ss.clear();
        ss.str("");

        //  This is to find High of the day
        size_t pos_of_later = page.find("Later%3A+");
        size_t high_ld = page.find("+(" , pos_of_later);
        size_t high_rd = page.find("+F" , high_ld);
        string high = page.substr(high_ld + 2 , high_rd - high_ld - 2);
        double thigh = atof(high.c_str());
        cout<<"Today's␣High␣is␣"<<f2c(thigh)<<"␣C"<<endl;
        ss << f2c(thigh);
        hight -> value(ss.str().c_str());
        ss.clear();
        ss.str("");

        //  This is to find Low of the day
        size_t low_ld = page.find("%2F+" , pos_of_later);
        size_t low_rd = page.find("F)" , low_ld);
        string low = page.substr( low_ld + 4 , low_rd - low_ld - 5);
        double tlow = atof(low.c_str());
        cout<<"Today's␣Low␣is␣"<<f2c(tlow)<<"␣C"<<endl;
        ss << f2c(tlow);
        lowt -> value(ss.str().c_str());
        ss.clear();
        ss.str("");

        // This finds the current humidity level
        size_t pos_of_pwsrtb = page.find("pwsrt_b");
        size_t h_ld = page.find("value=" , pos_of_pwsrtb);
        size_t h_rd = page.find("><" , h_ld);
        string humidity_now = page.substr( h_ld + 7 , h_rd - h_ld - 8 ) + "␣%";
        oss << humidity_now;
        cout<<"Today's␣Humidity␣Level␣is␣"<<humidity_now<<"␣%"<<endl;
        humidity -> value(oss.str().c_str());
```

```
oss.clear();
oss.str("");


// This finds the current weather
size_t curw_ld = page.find("%3A+");
size_t curw_rd = page.find("+(" , curw_ld);
string curw = page.substr( curw_ld + 4  , curw_rd - curw_ld - 4 );
oss << curw;
cout<<"The current weather is "<<curw<<endl;
weather -> value(oss.str().c_str());
oss.clear();
oss.str("");
};
}

Fl_Output *weather=(Fl_Output *)0;

Fl_Output *temperature=(Fl_Output *)0;

Fl_Output *humidity=(Fl_Output *)0;

Fl_Round_Button *farenheit=(Fl_Round_Button *)0;

Fl_Round_Button *celsius=(Fl_Round_Button *)0;

Fl_Output *hight=(Fl_Output *)0;

Fl_Output *lowt=(Fl_Output *)0;

int main(int argc, char **argv) {
  { Window = new Fl_Double_Window(469, 568, "Weather");
    Window->align(65);
    { new Fl_Box(220, 244, 30, 16);
    } // Fl_Box* o
    { input2 = new Fl_Input(159, 43, 150, 22, "Zip Code:");
    } // Fl_Input* input2
    { calculate = new Fl_Button(316, 45, 45, 20, "OK");
      calculate->callback((Fl_Callback*)cb_calculate);
    } // Fl_Button* calculate
    { weather = new Fl_Output(135, 478, 105, 24, "Weather:");
    } // Fl_Output* weather
    { temperature = new Fl_Output(135, 503, 105, 23, "Temp:");
    } // Fl_Output* temperature
    { humidity = new Fl_Output(135, 527, 105, 23, "Humidity:");
    } // Fl_Output* humidity
    { farenheit = new Fl_Round_Button(189, 69, 90, 15, "Fahrenheit");
      farenheit->type(102);
      farenheit->down_box(FL_ROUND_DOWN_BOX);
    } // Fl_Round_Button* farenheit
    { celsius = new Fl_Round_Button(189, 82, 90, 15, "Celsius");
      celsius->type(102);
      celsius->down_box(FL_ROUND_DOWN_BOX);
    } // Fl_Round_Button* celsius
    { new Fl_Output(205, 443, 65, 23, "Time␣until␣refresh:");
    } // Fl_Output* o
    { hight = new Fl_Output(282, 504, 105, 24, "High:␣");
    } // Fl_Output* hight
    { lowt = new Fl_Output(282, 478, 105, 24, "Low:␣");
    } // Fl_Output* lowt
    { Fl_Check_Button* o = new Fl_Check_Button(15, 15, 70, 20, "Mute?");
      o->down_box(FL_DOWN_BOX);
    } // Fl_Check_Button* o
    Window->end();
  } // Fl_Double_Window* Window
  Window->show(argc, argv);
  return Fl::run();
}
```

5 Test