

Computer Information Systems Portfolio

Anthony Galczak

WGalczak@gmail.com

Contents

Introduction	4
Technology Overview.....	5
Class Construction and Objects	6
C++	7
C#	9
Java.....	12
Python	15
Inheritance and Polymorphism.....	17
C++	18
Graphical User Interface	22
Java.....	22
Database Manipulation.....	26
Java.....	26
Game Development using Unity/C#	30
Web Research	39
Java.....	39
C	41
C	44

Introduction

This is a portfolio outlining the skills and technologies that I have learned while studying at Central New Mexico College from the summer of 2014 until my graduation in December 2016. I will graduate with an Associate of Applied Science in Computer Information Systems with a concentration in Computer Programming. My current GPA is a 3.9 as of the break before my final semester. I have also worked within the technical field throughout my studies at CNM.

I spent over 5 years working in technical support previous to enrolling at CNM. During my studies at CNM from early 2015 until 2016 I spent time as an IT consultant at Mintz Infotech where I assisted small businesses with general IT needs such as troubleshooting Mac OS X, Windows 7, 8 and administrating one Mac OS X server. During my time at Mintz Infotech I also was the primary writer and researcher of 3 books; Practical Paranoia: Android Security Essentials, Practical Paranoia: Android 5 Security Essentials and Practical Paranoia: Android 6 Security Essentials. This was a progressive project and the Android 6 version of the text was 646 pages of which I wrote over 600 pages myself. I also spent time writing a few excerpts (approx. 100 pages in total) for a Windows 10 book named Practical Paranoia: Windows Security Essentials. This set of texts is now being distributed to various technical colleges across the US for OS-specific security.

In my last few semesters at CNM, from September 2015 until current (Aug 2016 as of this writing), I have been a technical intern at Sandia National Laboratories. I spent the first 6 months at Sandia working in the job family of a "Solutions Architect" mostly doing Red Hat 7 Linux configuration management (using Ansible) as well as various deployed scripts mostly utilizing BASH and Python. The most recent period at Sandia has been spent on the MAUI software development team working on the near 2 million line code base of the MAUI project. In this latest role, I have been able to use my CNM education to write scripts in Python, analyze C++ code at a high level and even run code analysis tools such as Clang. This experience at CNM was invaluable towards being a very productive intern at Sandia and my career at Sandia will continue past my path at CNM. For the future, I will continue as an intern at Sandia and continue my studies at University of New Mexico for a Bachelor's of Science in Computer Science.

In this portfolio, you will see 4 programming languages showcased: C++, C#, Java, and Python. Within these languages I will also show the importance of object creation as well as utilizing inheritance and polymorphism. I have gained significant experience using these 4 languages at CNM and am proficient in developing production code for them. Through using these languages I have learned a significant amount of libraries that are too numerous to name that empower me to be an even stronger developer. There are also additional sections in this portfolio you will see such as implementing database logic and even a section on my additional game development and design course done in Unity and C#.

Technology Overview

CNM – Summer 2014 to Fall 2016

Relevant Coursework:

CIS 1250 - Python Programming I

CIS 1513 - Database Design/SQL

CIS 1680 - Linux Essentials

CIS 1275 - C++ Programming I

CIS 2520 - Introduction to SQL

CIS 1280 - .NET I/C Sharp

CIS 2275 - C++ Programming II

CIS 2521 - Database Programming with PL/SQL

CIS 2096 - Computer Game and Simulation Development

CIS 2235 - Java Programming I

CIS 2237 - Android Application Development with Java

CIS 2284 - .Net II

Languages: C++, C#, Java, Python, BASH, ASP.NET

IDEs: Notepad++, Visual Studio 2013/2015 (C++, C#), PyCharm, iDLE, Eclipse (Java), NetBeans (Java), Unity (C#), SceneBuilder, Android Studio

Version Control: Team Foundation Version Control

Libraries: wxPython, Swing, JavaFX, ASP.NET, LINQ, Android Java, WPF Forms

Sandia Labs – Sep. 2015 to Current

Relevant Classes:

Java 8 for Programming for OO Experienced Developers(TT2100-J8) including Java 8 Advanced Programming(TT3100-J8) – GlobalKnowledge – 07/25/16 - 07/29/16

Advanced Python for Scientists and Engineers – Enthought – 08/02/16 - 08/04/16

Languages: Python, Java, C++, BASH

Basic Editor Familiarity: vim, Notepad++

IDEs: Eclipse (C++ and Python), Canopy (Python)

Version Control: AccuRev, git

Libraries: Cython, Enthought Canopy Scientific suite

Class Construction and Objects

Using classes and objects is a fundamental piece of Object Oriented Programming. This infrastructure is necessary to abstract away otherwise difficult concepts into a concise package called a class. If you want to add 10 students to a database for a school; rather than writing descriptions for the students each time you can create a class called Student. When you instantiate this class in the general format **Student john = new Student();** you have created a new object. Creating new objects allows for enormous flexibility in your design of the code. Rather than working with fields and methods you can work with completely abstracted objects. Additionally, when using polymorphism, you can make small modifications to the class and create a completely new object without going through the effort of rewriting the whole class. Thinking in this object-oriented way at first is very difficult, but this skill pays off in the end with simplifying and strengthening your problem-solving skills.

During my schooling at CNM I have created an enormous number of classes in C++, Java, Python and C#, but I will just describe just one in detail. In my first CIS class at CNM (Python Programming) I utilized a class called GeoPoint that defaulted its fields using a `__init__` class using self to reference these fields. Then from there this class includes methods for calculating distance as well as properties to access the various fields found within the class. This class was instantiated using its default constructor into an object later within the “Submit button handler” that generated new GeoPoint objects from the data found in a SQLite3 database. Then the user data is captured from the wxPython GUI to generate another GeoPoint object that will be used to compare against the new list of GeoPoint objects.

C++

CIS 2275 - C++ II – Fall 2015

Program 3 – NameSearch Class

This C++ application is able to parse a .txt or .csv-like file and search individually for first or last names. There is logic built within the program to separate first and last names and sort them alphabetically.

Example usage of this program:

```
What file would you like to use?(Please enter a number)
1.)Enter my own file
2.)Use default P3_Names10K.txt
2

Your file was read successfully.
10000 names read.

1.) Search for a first name
2.) Search for a last name
3.) How many items are in the array?
4.) Exit the program
2

Please enter a last name:
manuel
SEARCHING FOR LAST NAME: MANUEL

Success! Your name was found.
We found 2 names.
EMANUEL, TERESA
MANUEL, MARY

1.) Search for a first name
2.) Search for a last name
3.) How many items are in the array?
4.) Exit the program
```

Constructor declarations in NameSearch.cpp:

```
NameSearch::NameSearch()
{
    // Initializing names array to nothing, initializing other variables to nothing
    for (int i = 0; i < 10000; i++)
    {
        names[i] = "";
    }
    total = 0;
    name = "";
    filename = "";
}
```

Default constructor for NameSearch class

```

        bReady = false;
    }

    NameSearch::NameSearch(string fName)
    {
        // Initializing names array to nothing, initializing other variables to nothing
        for (int i = 0; i < 10000; i++)
        {
            names[i] = "";
        }
        total = 0;
        name = "";
        filename = "";
        bReady = false;

        // Using overloaded constructor to be able to read file using fName, assigning
        this into class variable filename
        filename = fName;

        // Calling method for reading the file
        ReadFile();
    }

```

Overloaded constructor with a string parameter for NameSearch class

Assigning the constructor parameter "fName" to the class field "filename"

Object declaration and usage in Driver.cpp:

```

// Instantiating object with default constructor
NameSearch name;

// If the user wants to use their own file then use their file for the setfilename
method
if (fileRef == 1)
{
    cout << "What would you like to name your file?(Remember your 3 digit
extension)\n";
    cin >> userFile;
    name.SetFileName(userFile);
}
// If the user wants to use the default file then set the file using the set file
method
else if (fileRef == 2)
{
    name.SetFileName(FILE_IN);
}

```

Object declaration

Using a method on the newly declared object "name"

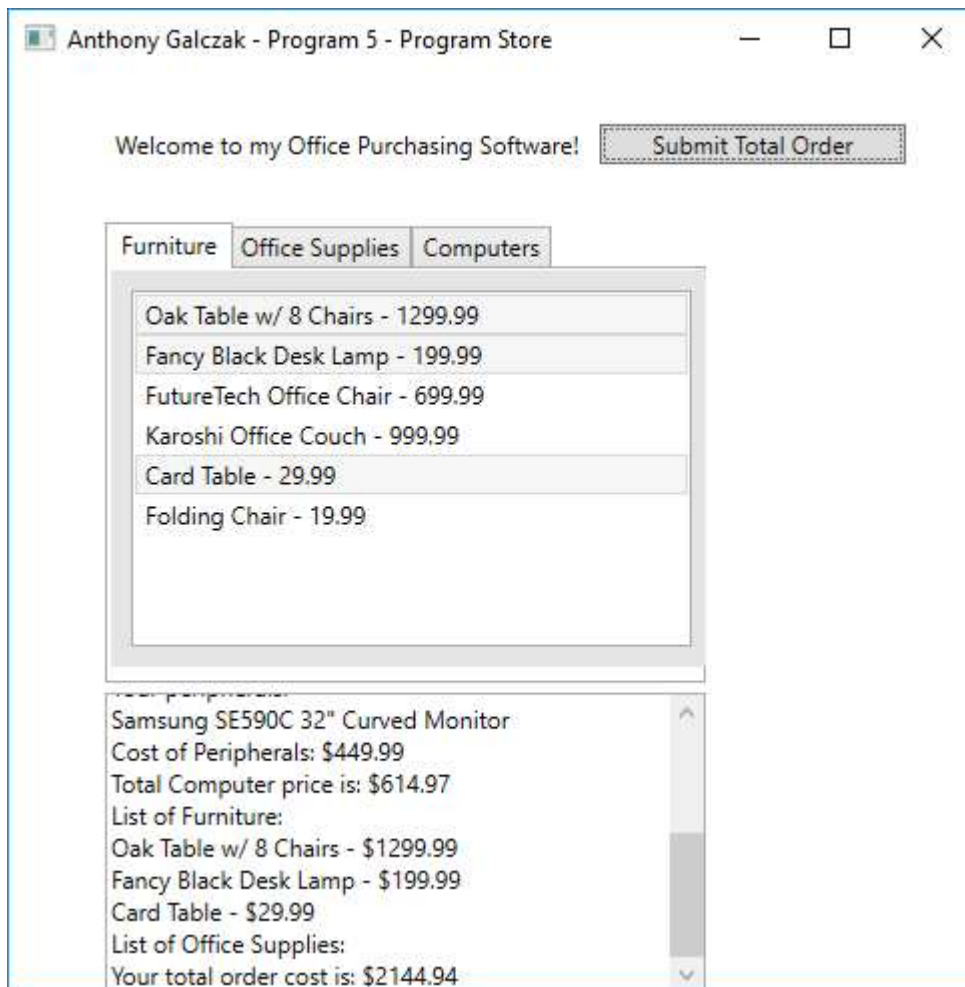
C#

CIS 1280 - .NET I/C Sharp – Fall 2015

Program 5 – Store on a Form including Computer Order Program

This C# application allows you to select items for a store. An initial frame opens with options to purchase furniture and office supplies. There is a further dialog to open the ComputerOrderForm which contains the previous program's frame to order a computer. It uses the Linq library which uses a XAML front-end to display the UI. I have also written a custom exception (ItemNotAvailableException) for this class that is intended to be thrown if an item is not available via a database.

Example usage of this program:



Anthony Galczak - GalczakP4 - Computer Store on a Form

Welcome to the C# Computer Store. Enter the following information to order a computer.

Select a Motherboard Combination:

Gigabyte Z97 + i7-4790k - 484.99

Select a Drive Option:

☐ Western Digital Blue 1TB - 54.99
☐ Seagate 2TB - 111.99
☒ Samsung 1TB SSD - 369.99

Select a Memory Option:

☐ Silicon Power 8GB - 54.99
☐ Crucial Ballistic Sport XT 16GB - 89.99
☒ Kingston Beast 32GB - 160.49

Select Peripherals:

☒ Samsung SE590C 32" Curved Monitor - 449.99
☐ Logitech G700s Wireless Gaming Mouse - 89.99
☒ Logitech G710+ Mechanical Gaming Keyboard - 139.99
☐ Bose SoundLink Wireless Headphone II - 279.95
☒ LG Super Multi Blue 16X Blu-Ray Burner - 64.99
☐ Key Tronic Budget Laser Mouse - 10.99
☐ Key Tronic Budget Keyboard - 10.99
☐ HP LV2311 23" LED Monitor - 99.99

Your motherboard: Gigabyte Z97 + i7-4790k
 Cost is: \$484.99
 Your memory: Kingston Beast 32GB
 Cost is: \$160.49
 Your drive: Samsung 1TB SSD
 Cost is: \$369.99
 Your peripherals:
 Samsung SE590C 32" Curved Monitor
 Logitech G710+ Mechanical Gaming Keyboard
 LG Super Multi Blue 16X Blu-Ray Burner
 Cost of Peripherals: \$0
 Total price is: \$1670.44

Submit **Clear**

Please enter information in form then click Submit.

Constructor declarations in ComputerOrder.cs:

```
// Constructors
public ComputerOrder()
    : this(1, 1, 1, new List<int>())
{ }

// Overloaded constructor
public ComputerOrder(int moboOptionIndex, int memoryOptionIndex, int
driveOptionIndex)
    : this(1, 1, 1, new List<int>())
{ }

// Even more overloaded constructor with initialization for variables from other
2 linked constructors
public ComputerOrder(int moboOptionIndex, int memoryOptionIndex, int
driveOptionIndex, List<int> periphList)
{
    this.moboOptionIndex = moboOptionIndex;
    this.memoryOptionIndex = memoryOptionIndex;
    this.driveOptionIndex = driveOptionIndex;
    this.periphList = periphList;
}
```

Default constructor for ComputerOrder class that is linked to the fully overloaded constructor

Overloaded constructor for ComputerOrder class that is linked to the next overloaded constructor

Fully overloaded constructor for ComputerOrder class that initializes all of the parameters to class fields

Object declaration and usage in ComputerOrderForm.xaml.cs:

```
public partial class ComputerOrderForm : Window
{
    private ComputerOrder comp;

    public ComputerOrder Comp
    {
        get { return comp; }
        set { comp = value; }
    }

    public ComputerOrderForm(ComputerOrder computerOrder)
    {
        InitializeComponent();

        comp = computerOrder;

        // Setting local arrays to keep the names of the fields from the
        ComputerOrder class
        string[] driveItems = comp.DriveInfo;
        string[] moboItems = comp.MoboInfo;
        string[] memItems = comp.MemoryInfo;
        string[] periphItems = comp.PeripheralsInfo;
    }
}
```

Object declaration

Encapsulating the ComputerOrder object within the form's class using an accessor and mutator

ComputerOrderForm constructor that initializes a ComputerOrder object

Declaring new string arrays within the ComputerOrderForm that are assigned via the ComputerOrder's info methods

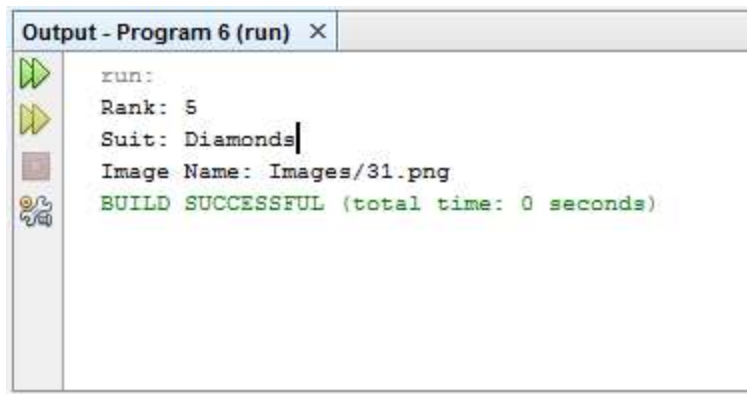
Java

CIS 2235 – Java Programming I – Spring 2016

Program 6 – ThreeCardBrag Program

This program is written in Java and was a group project between 3 students in this class. I wrote the object class Card that contains all of the methods to declare suit, rank and image. I also wrote the CardDeck class that populates a “deck” of Card objects (52 of them) to make up a brand new deck of cards to play a card game called Three Card Brag. I assisted the other team members with writing the game logic as well as the actual GUI form in JavaFX SceneBuilder. The end product was a card game that populates the form with Card objects via the CardDeck class.

Example execution of my driver testing the stand-alone classes Card and CardDeck:



Example usage of this program:



Constructor declarations in Card.java:

```
public class Card {  
    private int rank;  
    private String suit;
```

```
private String imageName;
```

```
public Card(){  
    rank = 0;  
    suit = "None";  
    imageName = "WhatImage";  
}
```

Default constructor for Card class

```
public Card(int r, String s, String i){  
    rank = r;  
    suit = s;  
    imageName = i;  
}
```

Overloaded constructor for Card class that uses 3 parameters to populate rank, suit and imageName

Object declaration and usage in CardDeck.java:

```
public class CardDeck {  
    private Card cardArray[] = new Card[52];  
    private boolean cardDealt[] = new boolean[52];
```

Declaring an array called cardArray containing 52 Card objects

```
    public CardDeck(){  
        // Initializing the cardDealt array to false  
        for(int i = 0; i < cardArray.length; ++i){  
            cardDealt[i] = false;  
        }
```

```
        // Initializing each spot of the array using the overloaded constructor  
        for(int i = 0; i < cardArray.length; ++i){  
            cardArray[i] = new Card(1, "Spades", "Images/1.png");  
        }
```

```
        // Loading cards with suit, rank, image  
        for(int j = 0; j < 4; ++j){  
            // 4 j's represent 4 sets of suits Ace-King  
            if(j == 0){
```

Initializing each Card object to a default value

```
                for(int k = 0; k < 13; ++k){  
                    cardArray[k].setSuit("Spades");  
                    cardArray[k].setRank(k+1); // Ranks start at 1, not 0  
                    String tempImageName = "Images/" + (k+1) + ".png";  
                    cardArray[k].setImageName(tempImageName);  
                }  
            }  
            else if(j == 1){  
                for(int k = 0; k < 13; ++k){  
                    int newK = k + 13; // New variable to exclude using a "magic number"  
                    cardArray[newK].setSuit("Hearts");  
                    cardArray[newK].setRank(k+1);  
                    String tempImageName = "Images/" + (newK+1) + ".png";  
                    cardArray[newK].setImageName(tempImageName);  
                }  
            }  
            else if(j == 2){  
                for(int k = 0; k < 13; ++k){  
                    int newK = k + 26;  
                    cardArray[newK].setSuit("Diamonds");  
                    cardArray[newK].setRank(k+1);  
                    String tempImageName = "Images/" + (newK+1) + ".png";  
                    cardArray[newK].setImageName(tempImageName);  
                }  
            }  
        }
```

```
    }  
    else{  
        for(int k = 0; k < 13; ++k){  
            int newK = k + 39;  
            cardArray[newK].setSuit("Clubs");  
            cardArray[newK].setRank(k+1);  
            String tempImageName = "Images/" + (newK+1) + ".png";  
            cardArray[newK].setImageName(tempImageName);  
        }  
    }  
}  
}
```

Python

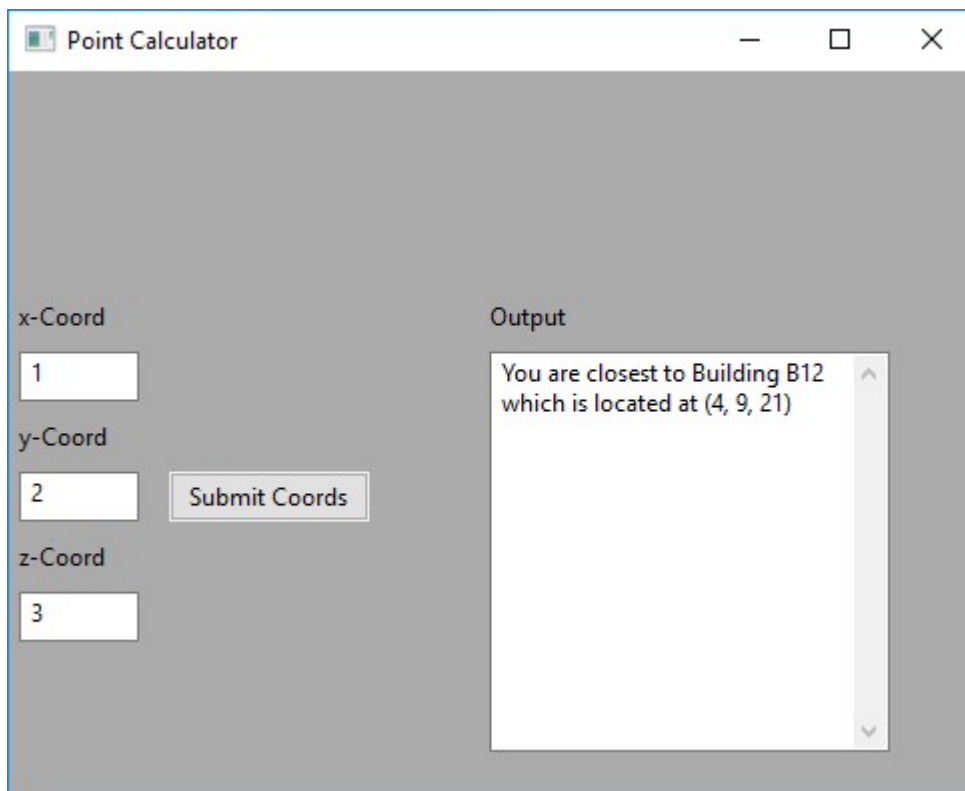
CIS 1250 – Python Programming I – Spring 2015

Program 12 – GeoPoint class with a Database

This program imports SQLite3 fields from a pre-populated database. With these fields imported it then generates a GeoPoint object with the information given in the database.

A user will then enter some coordinates that they would theoretically be standing at and the database will be polled to find out which 'Building' is the closest via a 3-D coordinate system.

Example usage of this program:



Constructor declarations in GalczakP12.py:

```
1. #---GeoPoint class---
2. class GeoPoint(object):
3.     def __init__(self, x=0, y=0, z=0, description = 'TBD'):
4.         self.x = x
5.         self.y = y
6.         self.z = z
7.         self.description = description
8.
9.     def SetPoint(self, coords):
10.        self.x = coords[0]
11.        self.y = coords[1]
12.        self.z = coords[2]
13.
```

Overloaded constructor declared for GeoPoint which also includes default parameter values for coordinates

```

14.     def GetPoint(self):
15.         return self.x, self.y, self.z
16.
17.     def Distance(self, toPoint):
18.         return sqrt(
19.             (self.x - toPoint.x)**2 +
20.             (self.y - toPoint.y)**2 +
21.             (self.z - toPoint.z)**2)
22.
23.     def SetDescription(self, description):
24.         self.description = description
25.
26.     def GetDescription(self):
27.         return self.description
28.
29.     PointCoords = property(GetPoint, SetPoint)
30.     PointDescription = property(GetDescription, SetDescription)

```

Object declaration and usage in GalczakP12.py submit button handler:

```

1.  #Database Interaction
2.      conn = sqlite3.connect('P12Points.db')
3.      curs = conn.cursor()
4.      curs.execute('SELECT * FROM Points')
5.      myTable = curs.fetchall()
6.  #Assigning individual indices
7.      for row in myTable:
8.          newX = row[0]
9.          newY = row[1]
10.         newZ = row[2]
11.         newDescription = row[3]
12. #Instantiating the point into GeoPoint
13.     newPoint = GeoPoint(newX, newY, newZ, newDescription)
14. #Appending pointList with Points
15.     pointList.append(newPoint)

```

Instantiating a new GeoPoint object with 4 parameters filled into the overloaded constructor

Inheritance and Polymorphism

Inheritance and polymorphism are both powerful mechanisms of Object-Oriented Programming. Inheritance is where you create a subtype or subclass of a class. For example; if you created a class **Person** and had fields **name, birthday, and address**. Now; you can inherit from the **Person** class via your subtype named **Employee**. In C++ and C# Person is called the base class and in Java it is called the super class. The power of inheritance shines in that you can add the method **Work()** to employee, but still have all the fields name, birthday, and address. This inheritance magic doesn't apply to just fields, you can also inherit classes with many methods as well. Not only does this save you the time of writing all the fields and methods that a person class contains, but it allows you to do clever designs. If you were to also create a subclass called **Student** that inherited from Person you could then add the method **Study()** to Student, but still retain the methods and fields of Person. From here you can now create an array of Person objects and fill it with your instantiated employees and students. Here's how this would look in C++:

```
// Creating the new student and Employee objects
Student anthony = new Student("anthony");
Employee robert = new Employee("robert");
Student sarah = new Student("sarah");

// Populating a Person array with Students and Employees
Person * people = new Person[3];
people[0] = anthony;
people[1] = robert;
people[2] = sarah;
```

Where this becomes truly polymorphic is if you were to write a for loop to iterate through each object in the people array and run the method **toString()** on them. In each implementation of Student and Employee the toString() method could differ. For example, the student toString may include their name, student ID and graduation date, but for the employee it would include their name, job title, and employee id. Because these two classes inherit from Person then Person would have to have a virtual method named toString() as well that is being overridden by each of these classes. This is the real power of polymorphism is being able to manipulate a set of objects that are fundamentally different but share enough at their base level that you can treat them as the same object.

One thing to keep in mind about polymorphism and inheritance is that it exists in all of the languages in this portfolio, but they are implemented with different syntax and nuances. When using object oriented design principles, it is extremely important to be able to utilize

inheritance and polymorphism correctly in order to avoid re-writing major segments of code and to separate your dependencies.

C++

CIS 2275 – C++ II – Fall 2015

Program 7 – Enigma Encoding Machine

This program implements a GUI that allows you to enter an encryption key via 1-50 or a randomly generated key, a phrase, and select one of three different encoding types. First, the program will bring the phrase/string through the Enigma base class via the Encode() method and then will send it through the inherited class' (Prime Shift, Shifty, or Qwerty) Encode() method.

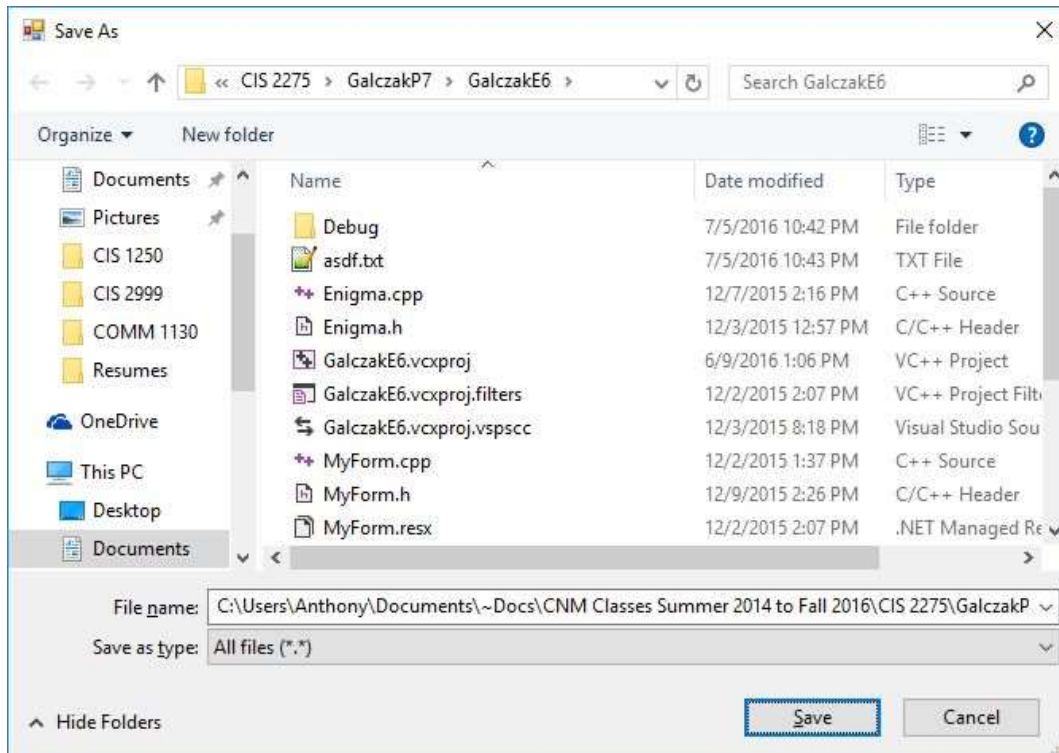
When selecting Encode a pop-up will appear so you can save a file in the directory of your choosing. This file will have the encrypted message, the key (via 1-50) and the type of encryption used (0-2).

The program can also read files that are outputted from this program again via the Encode button/method.

Example usage of this program:

The screenshot shows a Windows application window titled "Anthony Galczak - Program 7 - 3 Inherited Enigma Encodes". The window has a light gray background and contains the following elements:

- Welcome to the Enigma Encoding and Decoding Machine!**
Please enter your secret message to be encoded or decoded below:
- ENCODE: Choose One:**
 - ☒ Enter a Key: 26 (with a spin button)
 - ☐ Use Enigma Generated Key
- Encode Type:**
 - ☐ Prime Shift + Base Encode
 - ☒ Shifty + Base Encode
 - ☐ Qwerty + Base Encode
- Decode:** Just press the button!
This will open your secret file.
- Decode** button
- Clear** button
- Encode** button
- Secret Message Summary**
 - Message:** this is a test
 - Coded Message:** (%#(^%(\$%\$(\$@(^%\$(\$@*&&\$@(%#(^%\$(%#
 - Key:** 26



Virtual method declarations in Enigma.h:

```
#ifndef _ENIGMA_H
#define _ENIGMA_H

class Enigma
{
protected:
    string message;
    string codedMessage;
    int key;
    void Encode();
    void Decode();

public:
    Enigma();
    virtual void SetMessage(string mess, int k);
    virtual void SetMessage(string mess);
    virtual void SetCodedMessage(string codedM, int key);
    string GetCodedMessage(){ return codedMessage; }
    string GetDecodedMessage(){ return message; }
    int GetKey(){ return key; }
};

#endif
```

Virtual method declaration for setMessage

SetMessage method description in Enigma.cpp:

```
void Enigma::SetMessage(string mess, int k)
{
    // Check that key is between 1-50, if not set key to 1
}
```

```

    if (k > 50 || k < 1)
    {
        key = 1;
    }

    // Setting class variables
    message = mess;
    key = k;

    // Encoding the message we just received with the key
    Encode();
}

```

← Calling base class Encode method

Inherited class method declarations in QwertyEnigma.h:

```

class QwertyEnigma : public Enigma
{
public:
    QwertyEnigma();
    void SetMessage(string m, int k);
    void SetMessage(string m);
    void SetCodedMessage(string m, int k);

private:
    void Encode();
    void Decode();
};

```

← QwertyEnigma class inherits from Enigma base class

Inherited SetMessage method call in QwertyEnigma.cpp:

```

void QwertyEnigma::SetMessage(string m, int k)
{
    // Setting the message into the enigma base class
    Enigma::SetMessage(m, k);

    // Running the qwerty enigma encode
    Encode();
}

```

← Calling the base class' SetMessage method with the parameters m and k

← Calling QwertyEnigma's Encode method

Building array of inherited Enigma classes in MyForm.h:

```


// Making an array of Enigma-type pointers
Enigma *pE[3];

// Creating 3 children classes to be assigned into the array
PrimeShiftEnigma pse;
ShiftyEnigma se;
QwertyEnigma qe;

public ref class MyForm : public System::Windows::Forms::Form
{
public:
    MyForm(void)
    {
        InitializeComponent();
    }
}

```

```
    // Assigning the 3 child classes into the array
    pE[0] = &pse;
    pE[1] = &se;
    pE[2] = &qe;
}
```



Populating the Enigma array called pE with references to subclasses of pse, se and qe

Graphical User Interface

GUIs are extremely important for the functionality of a computer program. Even though most developers are very comfortable in a console (I personally have a love for vi terminals all over at work), it is highly necessary to present a graphical interface to users to be able to use your code. In order for a user to properly utilize your program there needs to be a “flow” to your GUI that intuitively guides them through using your program.

One of the things I value the most about writing GUIs is the ability to catch exceptions via design. Of course you want to be catching proper exceptions that could possibly blow up your program but if you don’t want your user to enter a number over 50 then you can limit your dialog box to only accept numbers 1 through 50. I really believe this is a very powerful point to keep in mind when writing your GUIs to be able to save the user from themselves.

I have written functional GUIs in Java, C++, C#, and Python.

Java

CIS 2235 – Java Programming I – Spring 2016

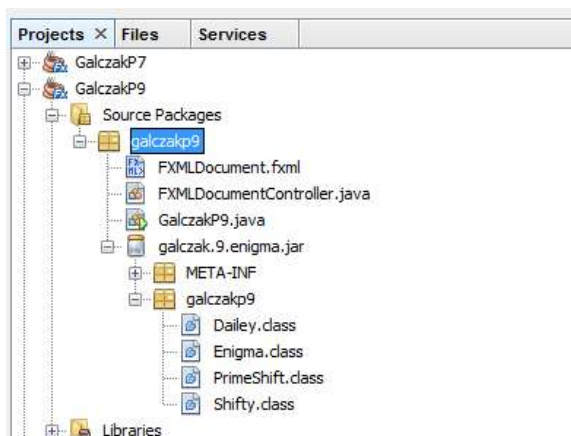
Program 9 – Enigma Encoding Machine

This program is very similar to the C++ version of it that I showed in the polymorphism section. In this java-written version of the program there is a small file I/O drop down that allows you to select a file to save or open when either encoding or decoding.

A quick summary of the enigma encoding program: You write a secret message and are able to save it in a text file. This file can be opened later on via the enigma encoder/decoder and decoded from the gibberish message that is saved.

This UI was written in Java using JavaFX and Scenebuilder.

This program also utilized creating a library from our previous program (Program 8) as a .jar file and importing it into this program for use. The Enigma, Prime Shift, Shifty, Dailey classes are all part of this jar.



Example usage of this program:

The application window has a title bar with standard Windows controls. Below the title bar is a menu bar with 'File' and 'Help'. The 'File' menu is open, showing 'Save File' and 'Open File'. The main area has a heading 'Welcome to the Enigma Encoding and Decoding Machine!'. It contains two columns of instructions: the left column explains encoding (enter message, select key, press Encode, save file), and the right column explains decoding (select Open File, browse to file, press Decode). Below the instructions are three buttons: 'ENCODE' (highlighted with a blue border), 'DECODE', and 'CLEAR'. There are two radio buttons for key selection: 'Use an Enigma-Generated Key' and 'Enter a Key, 1-50' (selected). A text box next to 'Enter a Key' contains the value '22'. Below these are three radio buttons for the encoding method: 'Prime Shift', 'Shifty', and 'Dailey' (selected). A section titled 'Secret Message Summary' contains a 'Message:' label and a text box with 'This is a secret message'. Below that is a 'Coded Message:' label and a text box containing a complex alphanumeric string. At the bottom are 'Key:' and 'Index:' labels, each followed by a text box containing '22' and '2' respectively.

File Help

Save File
Open File

Welcome to the Enigma Encoding and Decoding Machine!

To Encode, enter your message in the Text Box below
Select Enter a Key or Enigma-Generated Key
then press the Encode button below.
Select File > Save File to save your message to a file.

To Decode, Select File > Open File
then browse to your file to read the
coded message.
Press the Decode button below.

☐ Use an Enigma-Generated Key

☒ Enter a Key, 1-50

Encoding Method: ☐ Prime Shift ☐ Shifty ☒ Dailey

ENCODE **DECODE** **CLEAR**

Secret Message Summary

Message:

Coded Message:

Key: Index:

The window is titled 'Save a Coded Message in a File'. It shows a file explorer view of a folder named 'CIS 2235' within a parent folder 'CIS 2235'. The left sidebar shows the navigation pane with 'This PC' selected. The main area displays a list of folders and files. The 'File name' field at the bottom contains 'secretMessage.txt' and the 'Save as type' dropdown is set to 'text files (*.txt)'. The 'Save' button is highlighted with a blue border.

Save a Coded Message in a File

Navigation: < > << >> << CNM Classes Summer 2... > CIS 2235 > Search CIS 2235

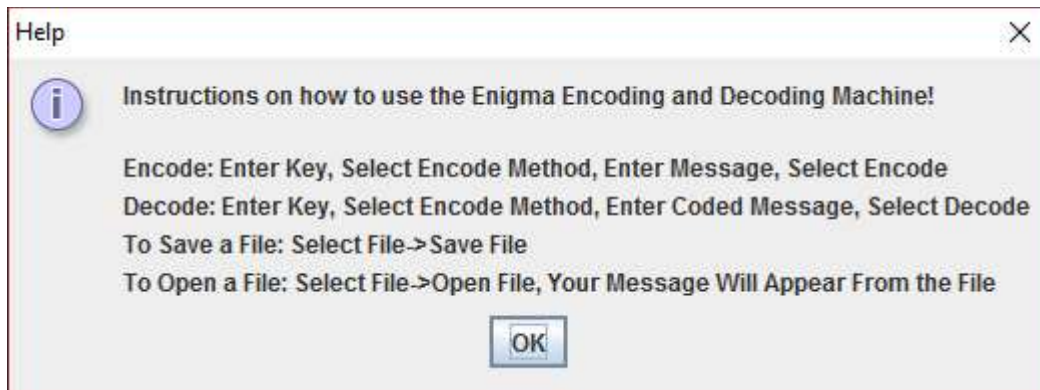
Organize New folder

Name	Date modified	Type
Galczak_P1	5/1/2016 8:27 PM	File folder
Galczak_P2	5/1/2016 8:27 PM	File folder
GalczakP3	5/1/2016 8:27 PM	File folder
GalczakP4	8/14/2016 8:38 PM	File folder
GalczakP5	5/1/2016 8:27 PM	File folder
GalczakP6	5/1/2016 8:27 PM	File folder
GalczakQ1	5/1/2016 8:27 PM	File folder
GalczakQ2	5/1/2016 8:27 PM	File folder
GalczakQ3	5/1/2016 8:27 PM	File folder
Loan Calculator Demo	5/1/2016 8:27 PM	File folder

File name:

Save as type:

Hide Folders **Save** Cancel



Initialize method in FXMLDocumentController.java:

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    radEnigmaKey.setSelected(true);
    txtEnterKey.setText("1");
    e[0] = pse;
    e[1] = se;
    e[2] = de;

    radPrimeShift.setSelected(true);
}
```

saveFileClick method in FXMLDocumentController.java:

```
@FXML
private void saveFileClick(ActionEvent event){

    //Create Filechooser object
    FileChooser fileChooser = new FileChooser();
    fileChooser.setInitialDirectory(new File("."));
    fileChooser.setTitle("Save a Coded Message in a File");
    //Set extension filter
    FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter(
        "text files (*.txt)", "*.txt");
    fileChooser.getExtensionFilters().add(extFilter);

    //Show the Save File Dialog
    File file = fileChooser.showSaveDialog(null);


    if(file != null)
    {
        PrintWriter outputFile = null;
        try {
            String filename = file.getCanonicalPath();
            File myFile = new File(filename);
            outputFile = new PrintWriter(filename);
            outputFile.println(e[index].getCodedMessage());
            outputFile.println(e[index].getKey());
            outputFile.println(index);

            outputFile.close();
        }
    }
}
```

Creating a save file dialog when
this method is executed.

Printing out the details of the coded
message to the file that we are saving


```
    } catch (IOException ex) {  
        Logger.getLogger(FXMLDocumentController.class.getName()).log(Level.SEVERE, null,  
ex);  
    }  
}
```



Logging a possible IOException error in case the file
doesn't write; generally results from permissions issues

Database Manipulation

Inevitably when coding up a CRUD interface for your data it will become useful to utilize a database. I have experience from CNM using Oracle SQL inside a JavaFX project, using SQLite3 in Python, LINQ in C#, as well as Oracle inside C++. All of these database interfaces allow you to store massive amounts of data outside of your code base. This modularizes your data from your actual logic and processing of this data. Databases additionally are designed to specifically be fast and efficient in the sort of everyday access that we use them for (CREATE, READ, UPDATE, DELETE).

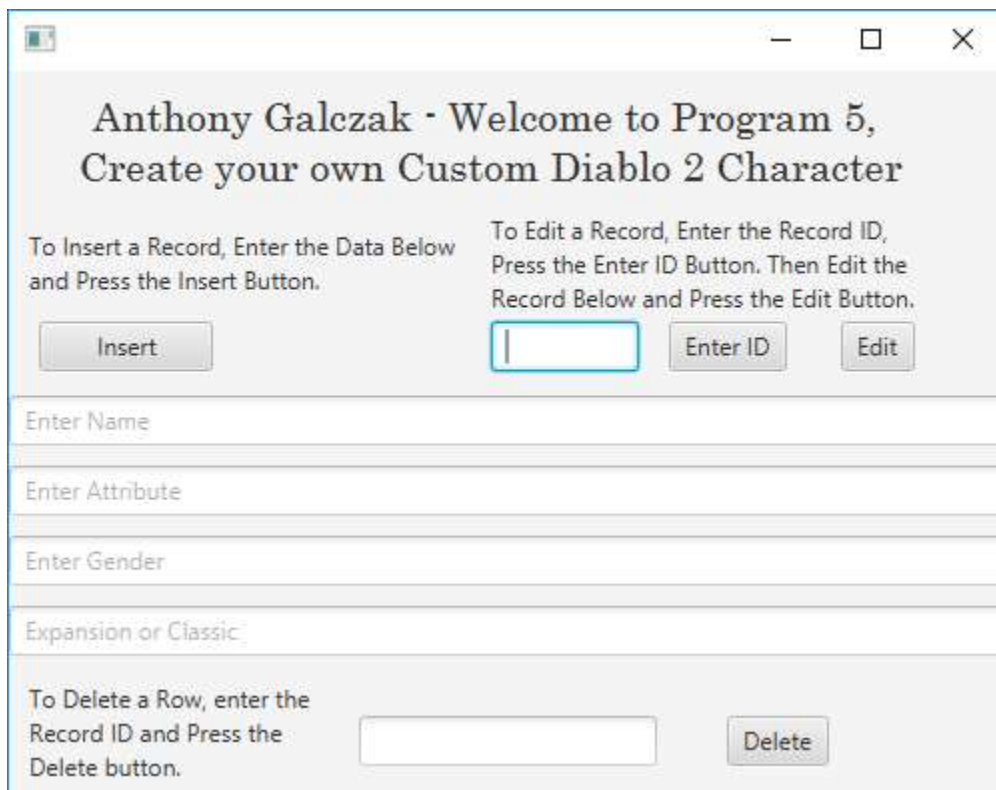
Java

CIS 2235 – Java Programming I – Spring 2016

Program 5 – Database

This program is a Java application that interfaces with an Oracle database to CRUD data in and out. I designed and wrote the class that populates the initial database with values as well as setting up a connection to the database. This Oracle database is stored locally on CNMs server. This code involves writing queries in SQL syntax and running them in the open connection. The executeUpdate method allows the command to be ran on the database that is initialized via the connection string.

Example usage of this program:



Anthony Galczak - Welcome to Program 5,
Create your own Custom Diablo 2 Character

To Insert a Record, Enter the Data Below
and Press the Insert Button.

To Edit a Record, Enter the Record ID,
Press the Enter ID Button. Then Edit the
Record Below and Press the Edit Button.

Insert

Enter ID

Edit

Enter Name

Enter Attribute

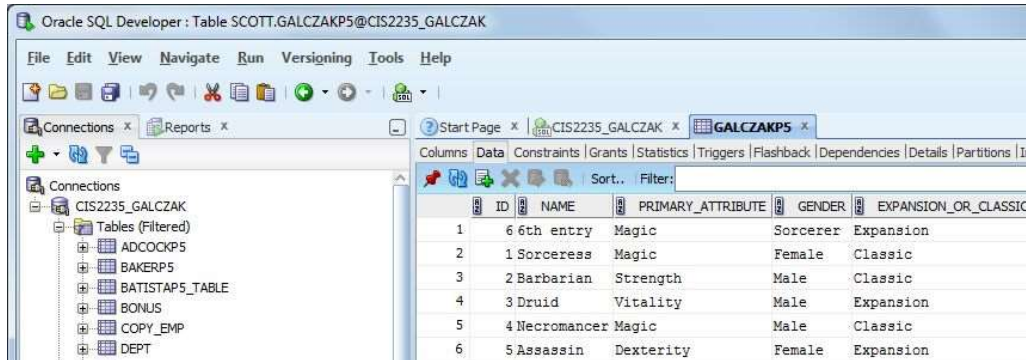
Enter Gender

Expansion or Classic

To Delete a Row, enter the
Record ID and Press the
Delete button.

Delete

View of the created database in SQL Developer:



The screenshot shows the Oracle SQL Developer interface. The left pane displays the 'Connections' tree with 'CIS2235_GALCZAK' expanded, showing a list of tables including 'GALCZAKP5'. The right pane shows the 'Data' tab for the 'GALCZAKP5' table, displaying a list of 6 rows with columns: ID, NAME, PRIMARY_ATTRIBUTE, GENDER, and EXPANSION_OR_CLASSIC.

ID	NAME	PRIMARY_ATTRIBUTE	GENDER	EXPANSION_OR_CLASSIC
1	6th entry	Magic	Sorcerer	Expansion
2	1 Sorceress	Magic	Female	Classic
3	2 Barbarian	Strength	Male	Classic
4	3 Druid	Vitality	Male	Expansion
5	4 Necromancer	Magic	Male	Classic
6	5 Assassin	Dexterity	Female	Expansion

Default constructor in dbManager.java:

```
//Constructor established the connection to the database
public dbManager() {
    try {
        // Load the JDBC driver
        Class.forName("oracle.jdbc.driver.OracleDriver");
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(dbManager.class.getName()).log(Level.SEVERE, null,
        ex);
    }

    try{
        // Establish a connection
        connection = DriverManager.getConnection
            ("jdbc:oracle:thin:@instora01.admin.ad.cnm.edu:1521:orcl","scott",
            "tiger");
        //statement = connection.createStatement();
    } catch (SQLException ex) {
        Logger.getLogger(dbManager.class.getName()).log(Level.SEVERE, null,
        ex);

        JOptionPane.showMessageDialog(null,"The database could not be
        located. " + "Please select the database"+ " file you wish to connect to.",
        "Database Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

Establishing a connection to the SQL database

createTable method in dbManager.java:

```
public void createTable() {
    try {
        statement = connection.createStatement();
        statement.executeUpdate
            ("Create table " + TABLE_NAME+ " " +
            "(ID NUMBER(3) NOT NULL PRIMARY KEY, "
            + "NAME Varchar2(30) NOT NULL, "
            + "PRIMARY_ATTRIBUTE Varchar2(30) NOT NULL , "
            + "GENDER Varchar2(10) NOT NULL , "
```

Method for creating the initial table with fields and datatypes

SQL statement "Create Table"

```

        + "EXPANSION_OR_CLASSIC Varchar2(30) NOT NULL )");
    } catch (SQLException ex) {
        Logger.getLogger(dbManager.class.getName()).log(Level.SEVERE, null,
ex);
    }
}

```

Insert method from dbManager.java (Create in CRUD):

```

public void insert(int recordID, String jname, String jattribute, String
jgender, String jexpacOrClass){
    try {
        //Insert a new record into the database
        String insertQuery =
            "INSERT INTO "+TABLE_NAME +
            " VALUES (" +recordID +", '"+jname+"', '"+jattribute+"',
            '"+jgender+"', '"+jexpacOrClass+"')";

        statement = connection.createStatement();
        statement.executeUpdate(insertQuery);
    } catch (SQLException ex) {
        Logger.getLogger(dbManager.class.getName()).log(Level.SEVERE, null,
ex);
    }
}

```

SQL statement "Insert". This reads in parameters that are given via the GUI to populate the SQL statement.

Executing the insert command via the SQL connection

getRecordById method from dbManager.java (Read in CRUD):

```

public String[] getRecordById(int recordID){

    String jname = "";
    String jattribute= "";
    String jgender = "";
    String jexpacOrClass = "";

    try {
        statement = connection.createStatement();

        //Display the contents of the record
        String newQuery = "SELECT * "
            +"FROM "+TABLE_NAME
            +" WHERE ID = " + recordID;
        ResultSet result = statement.executeQuery(newQuery);

        if(result.next())
        {
            jname = result.getString(2);
            jattribute = result.getString(3);
            jgender = result.getString(4);
            jexpacOrClass = result.getString(5);
        }
    }
}

```

SQL statement "Select". This grabs the row of data based on what recordID is given via arguments.

```

    } catch (SQLException ex) {
        Logger.getLogger(dbManager.class.getName()).log(Level.SEVERE, null,
ex);
    }
    String[] getRow = {jname, jattribute, jgender, jexpacOrClass};
    return getRow;
}

```

editRecord method from dbManager.java (Update in CRUD):

```

public void editRecord(int recordID, String jname, String jattribute, String
jgender, String jexpacOrClass){
    try {
        statement = connection.createStatement();
        //update the record
        String editQuery = "UPDATE " +TABLE_NAME
            + " SET NAME = '"+jname+"', PRIMARY_ATTRIBUTE =
 '"+jattribute+"', GENDER = '"+
            jgender+"', EXPANSION_OR_CLASSIC = '"+jexpacOrClass+"' "+
        "WHERE ID = "+recordID;
        statement.executeUpdate(editQuery);
    } catch (SQLException ex) {
        Logger.getLogger(dbManager.class.getName()).log(Level.SEVERE, null,
ex);
    }
}

```

SQL statement "Update". This updates the fields based on what is passed via arguments to the method.

deleteRecord method from dbManager.java (Delete in CRUD):

```

public void deleteRecord(int recordID) {

    try {
        statement = connection.createStatement();

        //Delete the record.
        String deleteQuery = "DELETE FROM " + TABLE_NAME + " WHERE ID = " +
recordID;
        statement.executeUpdate(deleteQuery);
    } catch (SQLException ex) {
        Logger.getLogger(dbManager.class.getName()).log(Level.SEVERE, null,
ex);
    }

}

```

SQL statement "Delete". This deletes the row based on a given recordID.

Game Development using Unity/C#

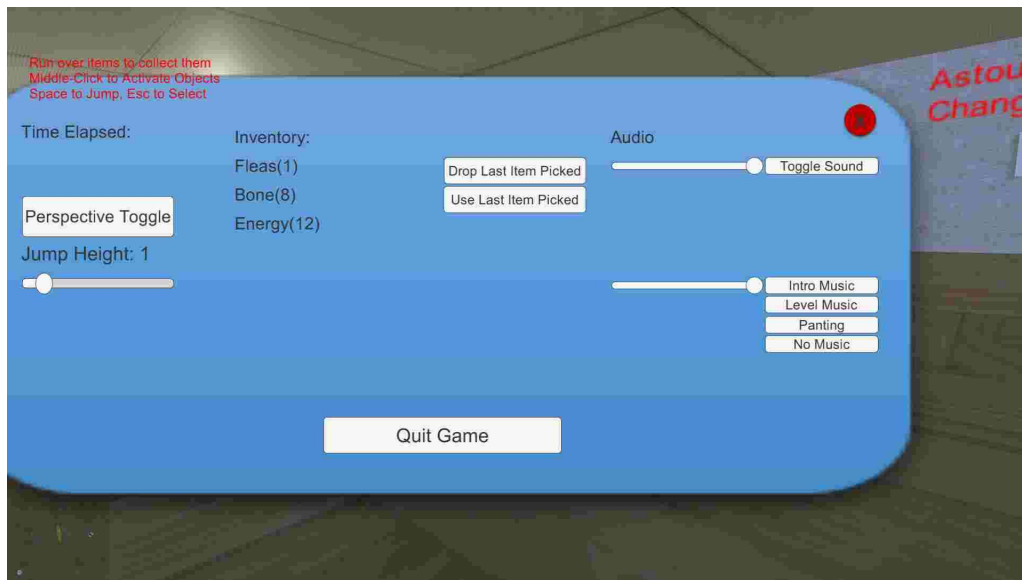
In the Spring 2016 semester a special topics class was being taught by Rob Garner called “Computer Game and Simulation Development”. This class was based in Unity using C# as a base language and Maya for 3-D modeling. and the class had a large variety of exercises and programs throughout the class. The class culminated with a group assignment using agile/scrum methodologies as well as an individual programming final project. Additionally, at the end of our class we utilized hardware interfaces for our games as a proof of concept. I used Google cardboard; an implementation of VR that uses a mount for your phone to create an immersive experience. I found that doing game development was one of the most difficult things I’ve done in CIS/CS and stretched the boundaries of thought when applying practical concepts to a game.

My individual game was designed to be a dog simulator. Jumping through sprinklers, knocking over boxes and jumping around in the yard was part of the standard fare. The game included picking up energy items in the form of spheres as well as 3-D model bones. These bones were collected to be able to advance to the next level (the yard). Within this game there was a functional clock, 3-D model chair made in Maya, a basic 3-D model of the dog player, destructible environment, AI characters who would stop when reaching an appropriate distance to the character. The number of features included in this game are astounding.

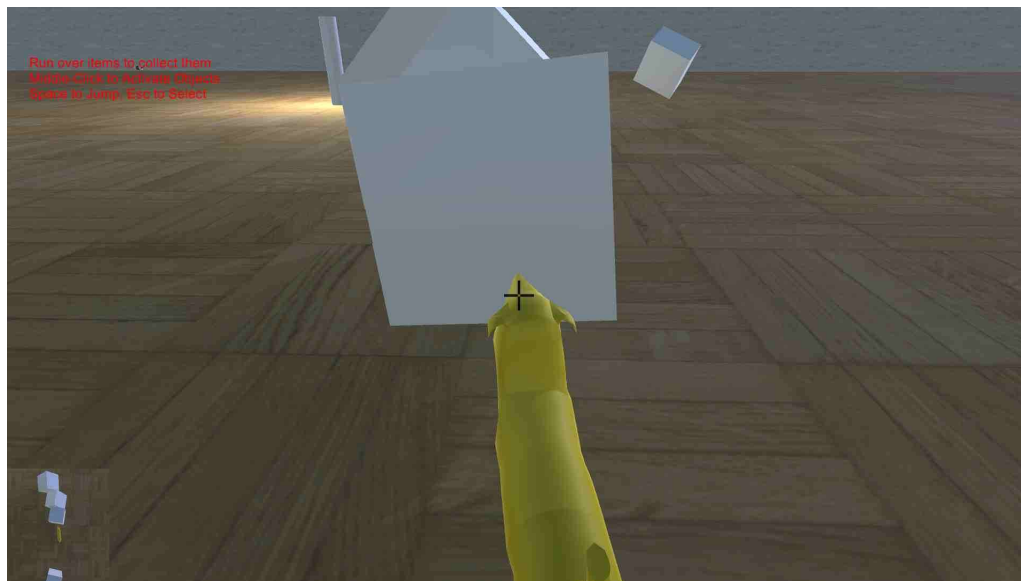
The group game was a first-person shooter that implemented a randomized level system. You could select wave mode or unlimited mode and you would be populated with a map chosen at random. These maps were pre-made prefabs that had the wall configurations setup already. A random int generator would then generate an int to pick one of these prefabs out of an array. The logistics of the bullets are very interesting as every time you press the trigger you must generate a bullet object that has its own weight and properties. These bullets were configured that upon impact of an AI character the NPC would die in one shot.

Individual Final Assignment screenshots:

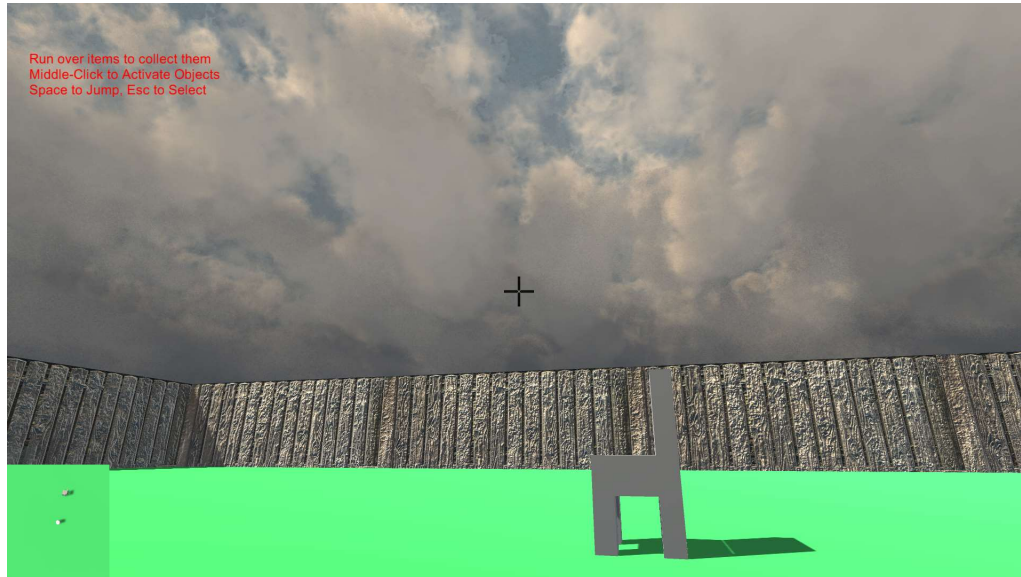
Interactive main menu that would allow you to toggle sound effects, music, sound effects volume, music volume, first-/third-person toggle, inventory system, a jump height modifier, item usage as well as quitting the game. This was a busy menu, but was intended to be a proof of concept for each element of a game that would be used. These are easily parsed out to button presses or an in-game menu. All of the items shown are functional.



This is an in-game screenshot of a third-person view of the game. This shows the destructible environment and boxes being thrown around by the main player character. Player movement was implemented with vectors moving from the origin on the map. This also shows the dog model that was implemented in Maya 3D. It has realistic movement including a model, skeleton, UV map, and texturing. The dog has moveable joints, that attach at foot, knee, and hip.



Here is the outside level. This map implemented a 3D skybox that wrapped the whole level. Hence the beautiful codes. You will also see a basic 3-D model chair made in Maya. The fence in the background is bump-mapped to appear more textured and realistic. The bump mapping was accomplished using a alpha-channeled map of the original texture skin and overlaying it on the original skin.



Here you can see part of the level design as well as an AI NPC. These NPCs roamed the map and will dodge the player character when they come within 5 meters of it. The level design included walls made from planes, as well as lighting objects implementing fixed lighting, sky-lighting and others.



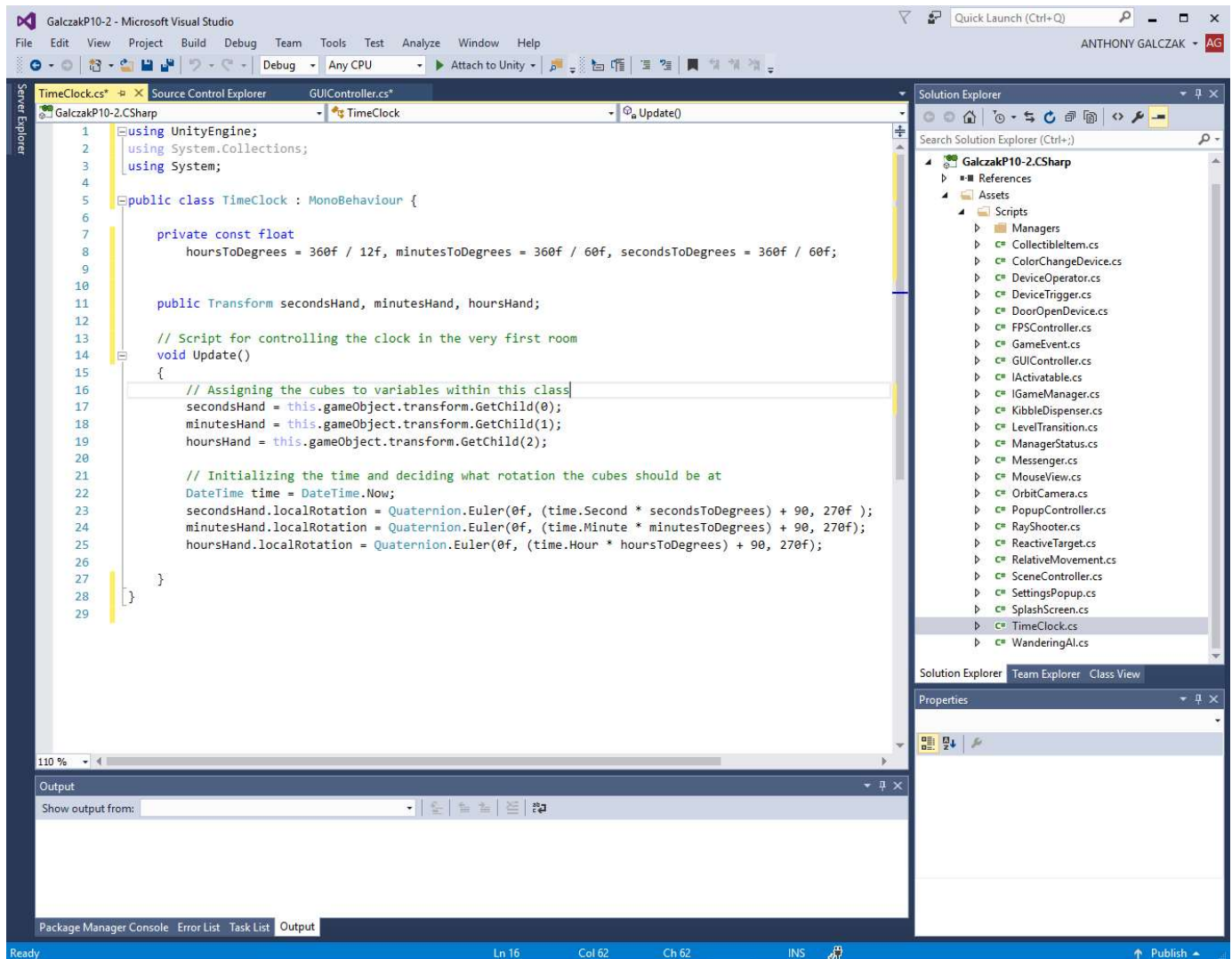
A view of the biggest part of the level. You can see a table and chairs to the right that were modelled in Maya. All textures for walls, ceiling, floors, etc. were brought in as Creative Commons textures. Off in the distance are some power-ups in the form of cubes and spheres that are used as “Health” and “Energy” pickups. When Energy pickups are used in the menu they will temporarily speed the dog character up 10x. This is done by increasing the speed modifier on the vector math that is done for player movement.



A functioning clock. This is done by rotating the clock's child objects(cylinders) using a transform method. The rotation on these cylinders is done using Quaternions to avoid kimbal lock if using these cylinders in multi-axis rotation.



TimeClock.cs from the Dog Simulator game:



This is the code explaining how the rotation on the clock works. There are interfaces in this game in the form of "IActivatable" (for doors and power-ups). There are classes handling movement, camera position, abstract classes for triggering events, level transition helper classes, AI logic, menu controls and a variety more.

RelativeMovement.cs from the Dog Simulator game:

```
// Update is called once per frame
void Update () {
    Vector3 movement = Vector3.zero;

    float horInput = Input.GetAxis("Horizontal");
    float vertInput = Input.GetAxis("Vertical");

    // Jump scripts
    bool hitGround = false;
```

Initializing a blank vector that will be modified for use with the Character Controller.

```

RaycastHit hit;

if (_vertSpeed < 0 && Physics.Raycast(transform.position, Vector3.down,
out hit))
{
    float check = (_charController.height + _charController.radius) /
1.9f;
    hitGround = hit.distance <= check;
}

if (horInput != 0 || vertInput != 0)
{
    movement.x = horInput * moveSpeed;
    movement.z = vertInput * moveSpeed;

    movement = Vector3.ClampMagnitude(movement, moveSpeed);
    Quaternion tmp = target.rotation;
    target.eulerAngles = new Vector3(0, target.eulerAngles.y, 0);
    movement = target.TransformDirection(movement);
    target.rotation = tmp;
    Quaternion direction = Quaternion.LookRotation(movement);
    transform.rotation = Quaternion.Lerp(transform.rotation, direction,
rotSpeed * Time.deltaTime);
}

if (hitGround)
{
    if (Input.GetButtonDown("Jump"))
    {
        _vertSpeed = jumpSpeed;
    }
    else
    {
        _vertSpeed = minFall;
    }
}
else
{
    _vertSpeed += gravity * 5 * Time.deltaTime;
    if (_vertSpeed < terminalVelocity)
    {
        _vertSpeed = terminalVelocity;
    }
    if (_charController.isGrounded)
    {
        if (Vector3.Dot(movement, _contact.normal) < 0)
        {
            movement = _contact.normal * moveSpeed;
        }
        else
        {
            movement += _contact.normal * moveSpeed;
        }
    }
}

```

Clamping movement speed
so that players do not move
faster on a diagonal

Allowing the player to jump via a
mapped key in Unity "Jump". In
game this ends up being Space.

```

    }
}

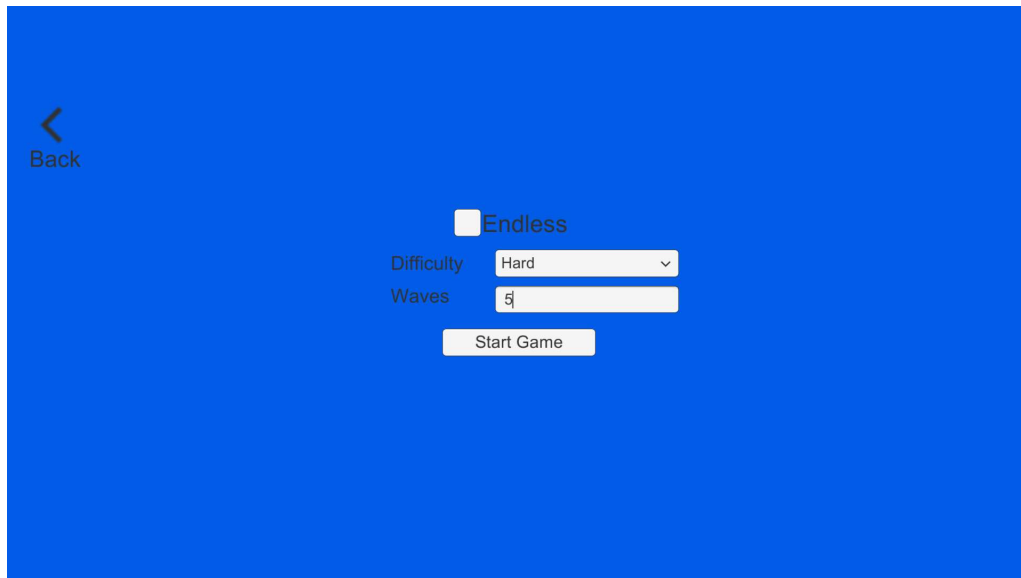
movement.y = _vertSpeed;
movement *= Time.deltaTime;
_charController.Move(movement);
}

```

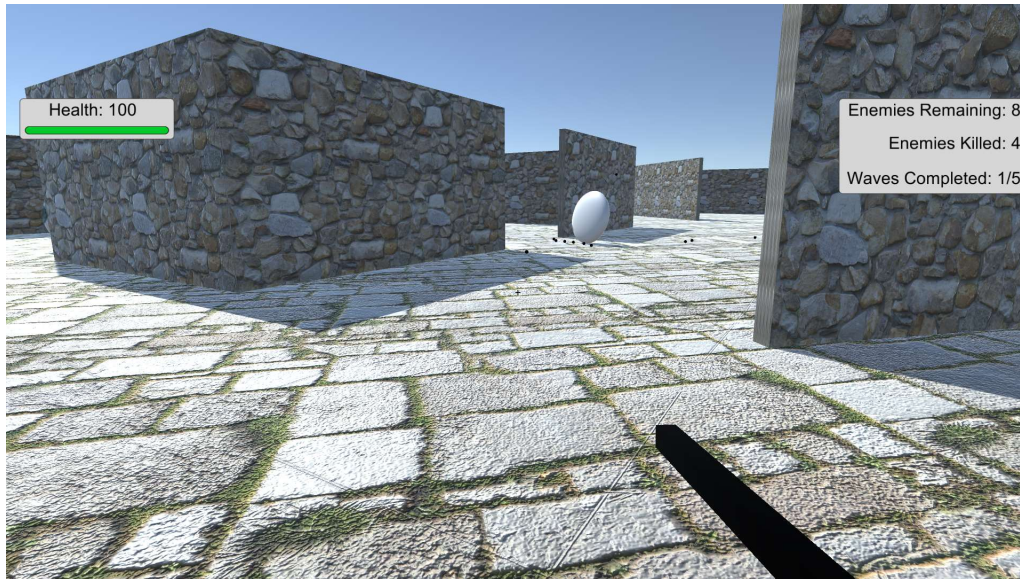
Moving the Character Controller in `deltaTime` so that it is smoothed for uneven framerates. Otherwise, higher frames would be faster movement.

Mega-Shooter 9000 Group Game Screenshots:

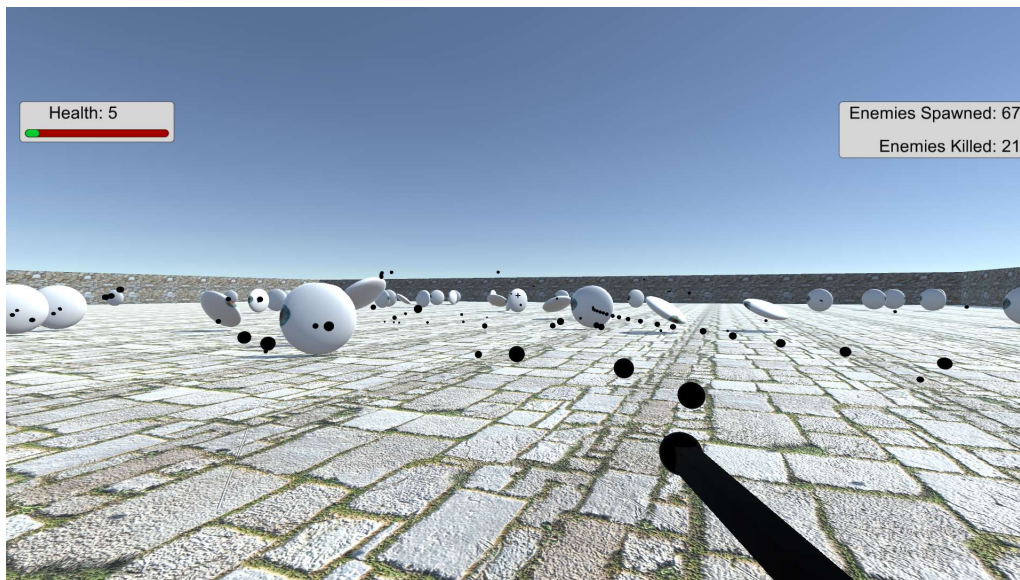
Interactive menu that allows you to select endless mode or instead how many waves you'd like to endure. It also includes a difficulty drop-down that increases the difficulty of the monsters from Easy to Medium to Hard.



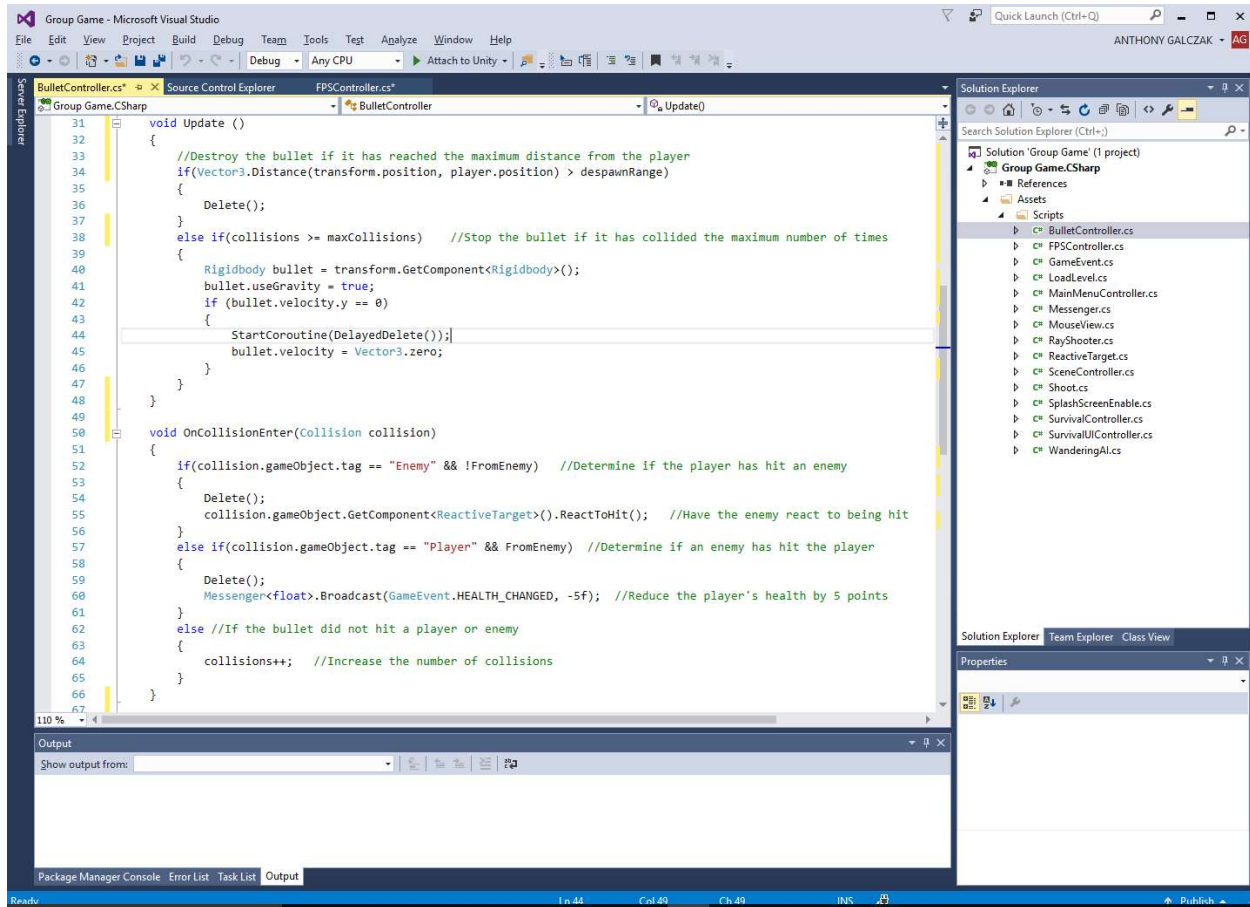
When playing wave mode, you will have a display box that gives you details about what wave you are on and how many enemies you have killed as well as how many have spawned.



Notice on endless mode how the menu has changed and I am being overwhelmed with enemies. Notice as well that the level has changed to a totally open level. This prefab is randomly selected upon game-spawn. This endless mode allows a fairly simple game mode to have an unbeatable, difficult challenge. I was only able to take the screenshot right before dying!



Part of BulletController.cs from Mega Shooter 9000:



This class is part of the bullet logic that is implemented in this game. This logic is where the game actually reacts to a bullet hitting a character. The ReactToHit method actually kills the AI character due to the bullet collision. The alternative is that the bullet is coming from an enemy and hitting a player. This then uses the messaging system to reduce the player's health by 5. Notice also that there is a variety of other classes in this game including player movement, shooting logic, and a variety of level-loading classes for the randomly generated levels.

Web Research

It is important to be able to use the web to research issues that you may be having with your code or to read the documentation on fresh libraries and ideas. With the age we live in, it is very difficult to survive without documentation on official resources such as Oracle's java docs or unofficial resources such as Stackoverflow.com. If someone has already solved a problem like yours then you can build upon the knowledge already built and implement this solution into your code. There is a reason that we don't write a fresh String class every time we are using Strings in Java; someone has already solved that problem with abstracting out the concepts for you.

Java

CIS 2237 – Android App Development – Fall 2016

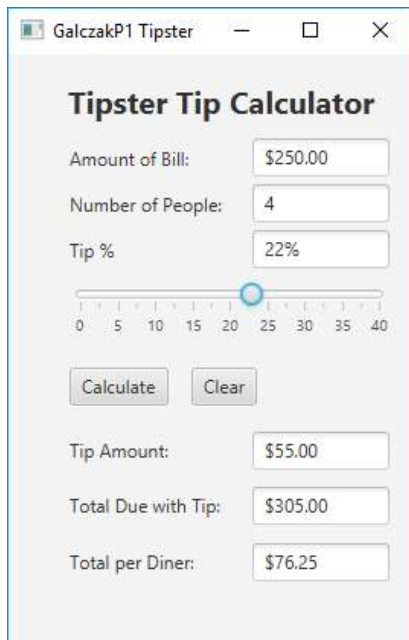
Program 1 – Tip Calculator

This program is an introductory review program designed to familiarize students with Java before moving onto the more advanced concepts found in the Android development portion of the course.

I made a tip calculator that will allow you to use a slider to adjust the tip in real-time after calculating the initial tip cost. This implementation is made using JavaFX and at the time I was a bit rusty on how to do an anonymous change listener for a Slider object.

My research consisted of going to Google and typing in "sliders in scenebuilder" and I was returned with many options, one of which being a JavaFX2 event handler and change listener tutorial. The link for this is <http://code.makery.ch/blog/javafx-2-event-handlers-and-change-listeners/>. I was able to parse this tutorial and include this anonymous ChangeListener into my own program and utilize it for my slider within my initialize method in FXMLDocumentController.java. Creating this anonymous ChangeListener allowed me to real-time modify the value of the slider in real-time which I was able to parse out to a "Model" class named TipsterCalc.java. Whenever this slider is "changed" or slid it will call the method setValues() that modifies all the values on the JavaFX view that the user sees.

Example usage of this program:



Tipster Tip Calculator

Amount of Bill:

Number of People:

Tip %:

Slider: 0 5 10 15 20 25 30 35 40 (Knob at 22)

Tip Amount:

Total Due with Tip:

Total per Diner:

Slider anonymous ChangeListener in initialize method in FXMLDocumentController.java:

```
// Slider construct obtained from
// http://code.makery.ch/blog/javafx-2-event-handlers-and-change-
listeners/
// Anonymous ChangeListener that executes upon a "changed" event
// Overriden changed method implements generic Number parameters
//
https://docs.oracle.com/javafx/2/api/javafx/beans/value/ChangeListener.html
slideTip.valueProperty().addListener(new ChangeListener<Number>() {
    public void changed(ObservableValue<? extends Number> observable,
        Number oldValue, Number newValue) {
        tc.setTipPercent(newValue.intValue());
        tfTipPercent.setText(np.format((tc.getTipPercent())));
    }
});

// If user has hit submit already, they can setValues via scrolling
if(submitted) setValues();
}
```


C

CS 241L – Data Organization in C – Fall 2016 - UNM

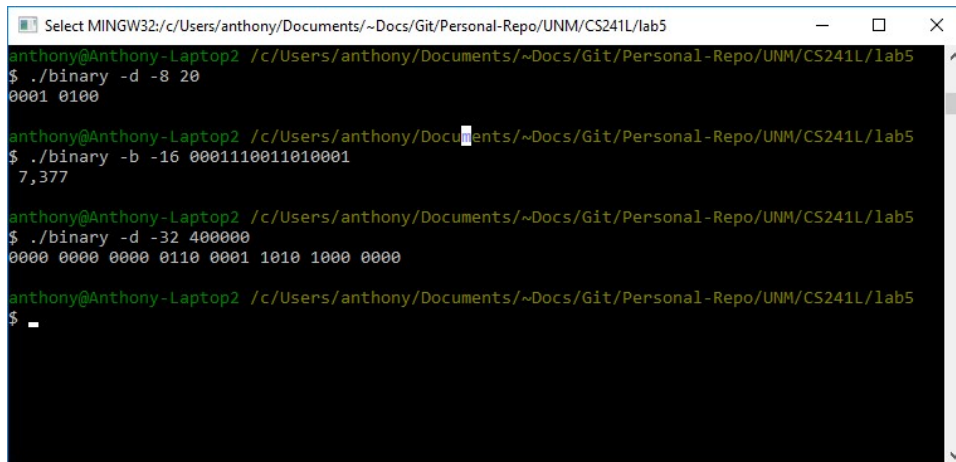
Lab 5 – Binary

This program is a bit unusual because it is from a lab assignment at UNM and it is also in the low-level language C. The binary lab is setup to convert a decimal number to binary and a binary number back to decimal. This is done using command line arguments a la argc and **argv parameters.

In my implementation of converting a number from decimal to binary I was using the method ltoa to use the binary base parameter to automatically convert a long number to an ascii character set. Unfortunately, as part of the ANSI standard this method is not included and therefore to be compliant needs to be included in my original source code.

I searched on Google, “ltoa source for c” and was returned with www8.cs.umu.se/~isak/snippets/ltoa.c. This site has a detailed source code for an ltoa implementation. I gained the permission of the TA and professor for the class and implemented this method into my binary.c.

Example usage of this program:

A screenshot of a terminal window titled "Select MINGW32:/c/Users/anthony/Documents/~Docs/Git/Personal-Repo/UNM/CS241L/lab5". The terminal shows the following commands and outputs:

```
anthony@Anthony-Laptop2 /c/Users/anthony/Documents/~Docs/Git/Personal-Repo/UNM/CS241L/lab5
$ ./binary -d -8 20
0001 0100

anthony@Anthony-Laptop2 /c/Users/anthony/Documents/~Docs/Git/Personal-Repo/UNM/CS241L/lab5
$ ./binary -b -16 0001110011010001
7,377

anthony@Anthony-Laptop2 /c/Users/anthony/Documents/~Docs/Git/Personal-Repo/UNM/CS241L/lab5
$ ./binary -d -32 400000
0000 0000 0000 0110 0001 1010 1000 0000

anthony@Anthony-Laptop2 /c/Users/anthony/Documents/~Docs/Git/Personal-Repo/UNM/CS241L/lab5
$ -
```

ltoa implementation in binary.c:

```
#define BUFSIZE (sizeof(long) * 8 + 1)

char *ltoa(long N, char *str, int base)
{
    register int i = 2;
    long uarg;
    char *tail, *head = str, buf[BUFSIZE];

    if (36 < base || 2 > base)
        base = 10; /* can only use 0-9, A-Z */
    tail = &buf[BUFSIZE - 1]; /* last character position */
}
```

```

*tail-- = '\0';

if (10 == base && N < 0L)
{
    *head++ = '-';
    uarg    = -N;
}
else uarg = N;

if (uarg)
{
    for (i = 1; uarg; ++i)
    {
        register ldiv_t r;

        r      = ldiv(uarg, base);
        *tail-- = (char)(r.rem + ((9L < r.rem) ?
                                ('A' - 10L) : '0'));
        uarg    = r.quot;
    }
}
else *tail-- = '0';

memcpy(head, ++tail, i);
return str;
}

```

decimalToBinary method in binary.c using ltoa:

```

/* Converting the number from decimal to binary */
void decimalToBinary(char* charNum, int charLen)
{
    long decimalOutput = 0;
    int numLength = 0, i = 0;
    char returnString[65];

    /* Error check for in case char array contains non-digits */
    while(charNum[i] != 0)
    {
        /* 48 == '0', 57 == '9', 0 == '\0' */
        if(charNum[i] < 48 || charNum[i] > 57)
        {
            printf("ERROR: argument 3 is not a decimal integer\n");
            printUsage();
            exit(0);
        }
        ++i;
    }

    decimalOutput = strtoul(charNum, &charNum, 10);

    /* Converting decimalNum to binary ascii representation */

```

```

ltoa(decimalOutput, returnString, 2);

/* How many digits are in the number */
numLength = strlen(returnString);

/* How many characters the user asked for */
charLen = abs(charLen);

for(i = 0; i < charLen; ++i)
{
    /* Adding a space every 4 digits */
    if((i % 4) == 0 && i != 0)
    {
        putchar(' ');
    }

    /* Deciding whether to add a leading 0 or output the real num
*/
    if(charLen > i + numLength)
    {
        putchar('0');
    }
    else
    {
        putchar(returnString[i - (charLen - numLength)]);
    }
}
printf("\n");
}

```

Calling ltoa with 3rd parameter
"base" set to 2 for binary.

Formatting the binary output with
spaces every 4 digits.

C

Lab 6 – Cipher

This program uses a cipher algorithm to encrypt or decrypt data given in a comma-separated form. I was attempting to capture the data that was given via stdin on the command line in the form “varvar,var,var\n”. This program has multiple .h and .c files, but the most important piece is that I can parse out the pre-formatted data and enter this data as parameters into my methods. There is also exception handling that occurs to check and make sure the data coming in is in the right type and size.

I have already captured the data on each line into a character array called line. I was unsure how to split the character array/string on a comma delimiter so I searched “reading a comma separated string in C”. The 3rd result was <http://stackoverflow.com/questions/15091284/read-comma-separated-input-with-scanf>. Given the context of how I have saved the data I needed to use sscanf to read the data out of the string that is already saved. The syntax that is given on stackoverflow.com is:

```
scanf("%4[^,],%4[^,],%79[^,],%d", sem, type, title, &value);
```

My code implements similar regex-style syntax with sscanf in the main method that reads in data from stdin.

```
// Saving strings for use in algorithm
sscanf(line, "%50[^,],%50[^,],%s\n", lcg_m, lcg_c, data);
```

This line parses out my variables that are sanitized and hopefully converted into integers for use in my encryption algorithm. Data is read into this algorithm one byte at a time.