24783 Advanced Engineering Computation: Problem Set 4

(*) In the following instruction (and in all of the course materials), substitute your Andrew ID for where you see *yourAndrewId*.

START EARLY!

1 Check Out or Update Base Code and Libraries

Please make sure you have up-to-date libraries and course files before starting an assignment.

If you have not done working-directory set up as described in the first assignment (like in case you need to work from a different computer), please see Problem Set 1 and set up the working directory.

I assume you created the working directory called *24783* under you home directory and you checked out your Git repository in there.

Home directory is typically *C:\Users\username* in Windows, */Users/username* in macOS, and */home/username* in Linux, where *username* is the user name in your local computer.

First, open command-line (Developer PowerShell or Terminal), and move to your working directory by typing:

```
cd ~/24783
```

You need to check out (or clone) Git repositories once. If you have not checked out yet, do the following:

```
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/course_files.git
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

You need to replace "yourAndrewId" with your Andrew ID. You'll be asked to type in credentials.

Also we are going to use two additional repositories:

```
git clone https://github.com/captainys/MMLPlayer.git
git clone https://github.com/captainys/public.git
```

If you are successful, you should have the following directory structure under your home directory.

```
Your User Directory
├── (Other files and directories)
└── 24783
    └── course_files
```

```
└── yourAndrewID
```

If you already have checked out these repositories (most likely you did for Problem Set 1), you need to update (or git pull) in those repositories. By change directory to the location where you checked out repositries and then type:

```
git pull
```

To update all four repositories, you can type the following commands in a sequence:

```
cd ~/24783/course_files
git pull
cd ~/24783/yourAndrewID
git pull
cd ~/24783/public
git pull
cd ~/24783/MMLPlayer
git pull
```

2 Copy Base Code and Add to Git's Control

Like ps1 through ps4, you can start from the base code. Copy ps4 directory to your working directory. I assume your working directory is:

```
~/24783
```

and, your copy is:

```
~/24783/ps4
```

After copying, your directory structure should look like:

```
Your User Directory
├── (Other files and directories)
└── 24783
    ├── course_files
    ├── public
    ├── MMLPlayer
    └── yourAndrewID
        └── ps4
            ├── ps4.pdf
            ├── ps4_1
            ├── ps4_2
            ├── hashutil
            ├── simplebitmap
            └── png
```

It is important that public, MMLPlayer, and yourAndrewID directories are at the same level, and ps4 directory is directly under yourAndrewID directory. Otherwise your program cannot be built in the grading environment.

3 Make a CMake Project

Write cmake scripts. You need to write one top-level script, and one each for a sub-directory (except png sub-directory). Therefore, you will write five CMakeLists.txt files in total.

The top-level CMakeLists.txt must add sub-directories public, MMLPlayer/ym2612, and MMLPlayer/mmlplayer. MMLPlayer is needed if you want to work on bonus problem 3.

Make sure to use target names, ps4_1, ps4_2, hashutil, and simplebitmap (same as the directory name, all small cases.)

4 Finish Missing Pieces in the Base Code

4.1 CutOut member function

Finish CutOut member function in the SimpleBitmapTemplate class. As it says, the purpose of this function is to copy a sub-region from the *this* bitmap. (I mean this-pointer by *this*.) The function takes a reference to the destination bitmap, top-left corner in the *this* bitmap, and the size of the sub-region to be cut out to the destination. The destination bitmap should be re-sized to the size of the sub-region. If the sub-region extends out of the *this* bitmap, r,g,b,a values (or all the components of the pixel) of the outside of *this* bitmap should be made clearColor parameter given to the CutOut function. See comment lines in simplebitmap.h for more detailed instruction.

4.2 Cut Out Tiles

In ps4_1/main.cpp, write the code that:

1. Reads a .PNG bitmap specified by the first argument (argv[1]) to the command.
2. Cut out 40x40-pixel blocks of the input PNG and save to individual .PNG files in the current working directory. The name of the PNG files must be 0.png, 1.png, 2.png, ... Output PNGs must be 40x40 regardless of the dimension of the input PNG. When the resolution of the input bitmap is not of 40*Nx40*M (N,M are integers), the last block in each row and column will have some transparent pixels. Stop writing after writing 200 PNGs or entire input PNG is covered, whichever happens first.
3. If the user does not provide the first argument, print a usage information as: "Usage: ps4_1 pngFileName.png"
4. If the program cannot read the .PNG image, print an error message as: "Error: Failed to read a .PNG file."
5. For this assignment, my PNG encoder will spit out a bunch of console output anyway. It is ok to leave some additional information if you need for debugging.

Then compile and run the program. You can use PNG files in course_files/ps4/png for testing.

Since your program is supposed to write many files to the current working directory, I suggest to run from the build directory. Once done, you can remove files by "rm *.png".

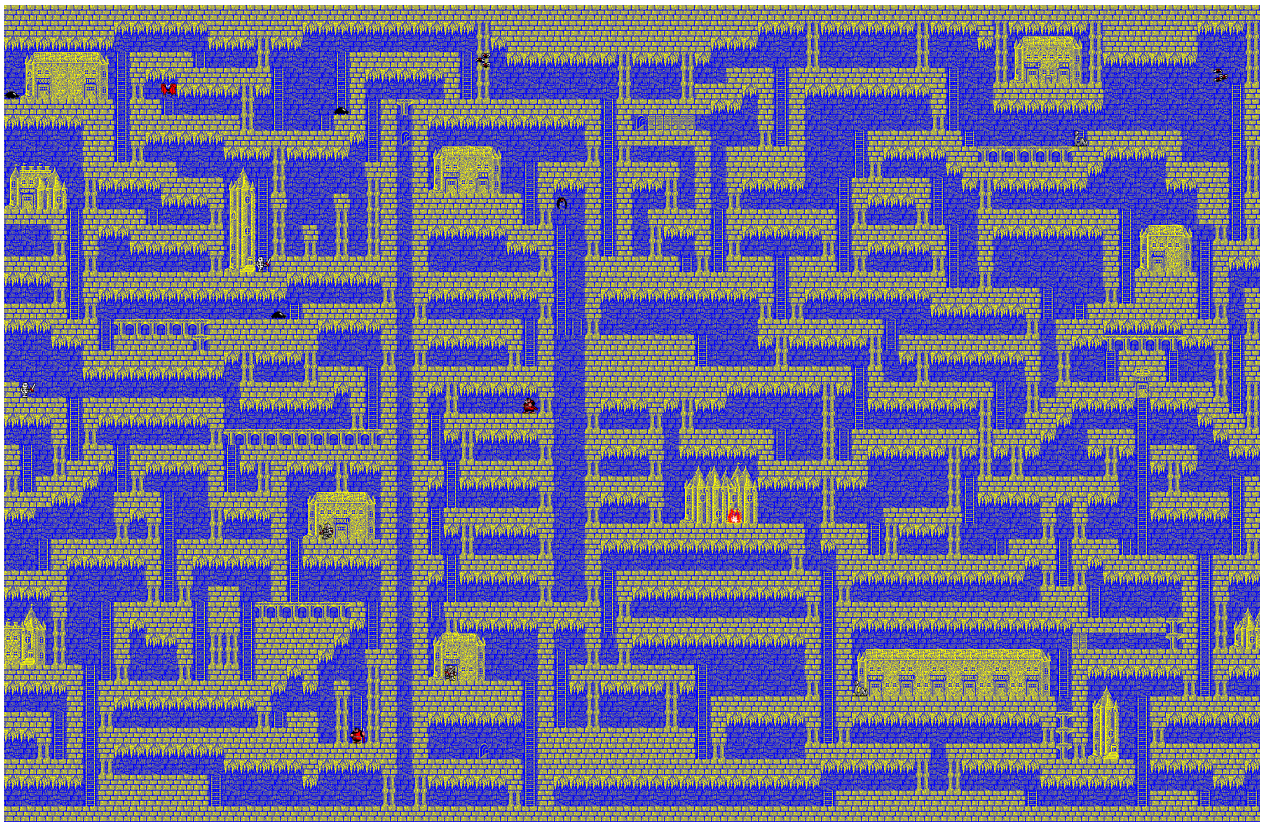

Fig. 1: Level 1 Map from Xanadu (c) Nihon Falcom 1985

5 How Many Bitmap Patterns in the Xanadu Map?

5.1 Complete autoResize_base Function in the ysHashBase Class

autoResize_base function in the ysHashBase class is left empty. Fill the function. resize_base function is already filled, and you can use the function.

5.2 Hash Table of Bitmaps

Fig. 1 is from a retro PC game called Xanadu by Nihon Falcom Inc., which was a huge success in 1985 in Japan. It sold in total 400K copies. Consider the number of PCs available in that day. You can imagine the magnitude of the popularity of this game. You can find images of the maps in course_files/ps4/png.

In the actual game, only 9x9 blocks are visible, but this map was generated by stitching screenshots together. I ended up wasting time during winter break a few years ago playing Windows port of this game and writing a program for stitching the screenshot simultaneously.

The question here is how many kinds of blocks this map is made of, and that is the goal of your program.

Your program in ps4_2 directory must take a PNG file name as input (first argument), and read it into SimpleBitmap data structure.

Then, your program must assign an ID number (an integer value) for each type of a 40x40 pixel block. To do so, cut out a 40x40 block, if an ID number is not given to the block pattern, assign a unique number, and

then add the block-number pair to the hash table. The ID number is what you find in the hash table. The 40x40 bitmap is a hash key.

The first block type should be given 0, and whenever your program finds a new type of a block increment the ID number.

You need to write code that:

1. makes a hash table that finds an integer value (ID number) from a bitmap.
2. specializes the HashCode function.
3. calculates a hash-code from a 40x40 bitmap.

For a hash code, what you essentially need to do it to calculate an integer value (do not confuse with an ID number) from an array of unsigned chars. There is no unique way of calculating a hash-code from a sequence of unsigned integers. You can come up with your own, or you may try the method described in:

```
http://stackoverflow.com/questions/11128078/android-compute-hash-of-a-bitmap
```

One thing I don't like about this approach is it will for sure let the hash code overflow during the calculation. What you get from overflow is undefined in C/C++ specification.

I rather would go with multiplying different prime number for different location in the array. Also, make it a non-4-byte cycle. Since all alphas of the map is 1.0, making a 4-byte cycle may make a pattern, which is not a good characteristic for a hash-code. For example, you may want to multiply:

- 2 to the (5*n)th bytes,
- 3 to the (5*n+1)th bytes,
- 5 to the (5*n+2)th bytes,
- 7 to the (5*n+3)th bytes,
- 11 to the (5*n+4)th bytes,

and add them up.

To make SimpleBitmap usable from the hash table, you need to define operator== and operator!=, and specialize ysHash for SimpleBitmap class. To compare bitmaps, you need to compare dimension and all RGBA values.

Then, draw blocks that your program finds on the window. The window size must be 1200x800. Draw blocks like tiles from top-left of the window. 30 blocks per row. The order can be arbitrary, but the same block must not be drawn twice. If this step is successful, your program show an image like Fig. 2 (when Level1.png is given as input).

Your program also need to output an ASCII-text representation of the map. One character for one 40x40 block. Line break after each row. Print ' '+ID number for the block. For example, do:

```
printf(\%c",' '+id);
```

I personally think it is easier to do it while you are assigning IDs to the blocks. It can be done by a few additional lines. If you are successful in this step, you will see an out put like Fig. 3 on the console window (when Level1.png is given as input).

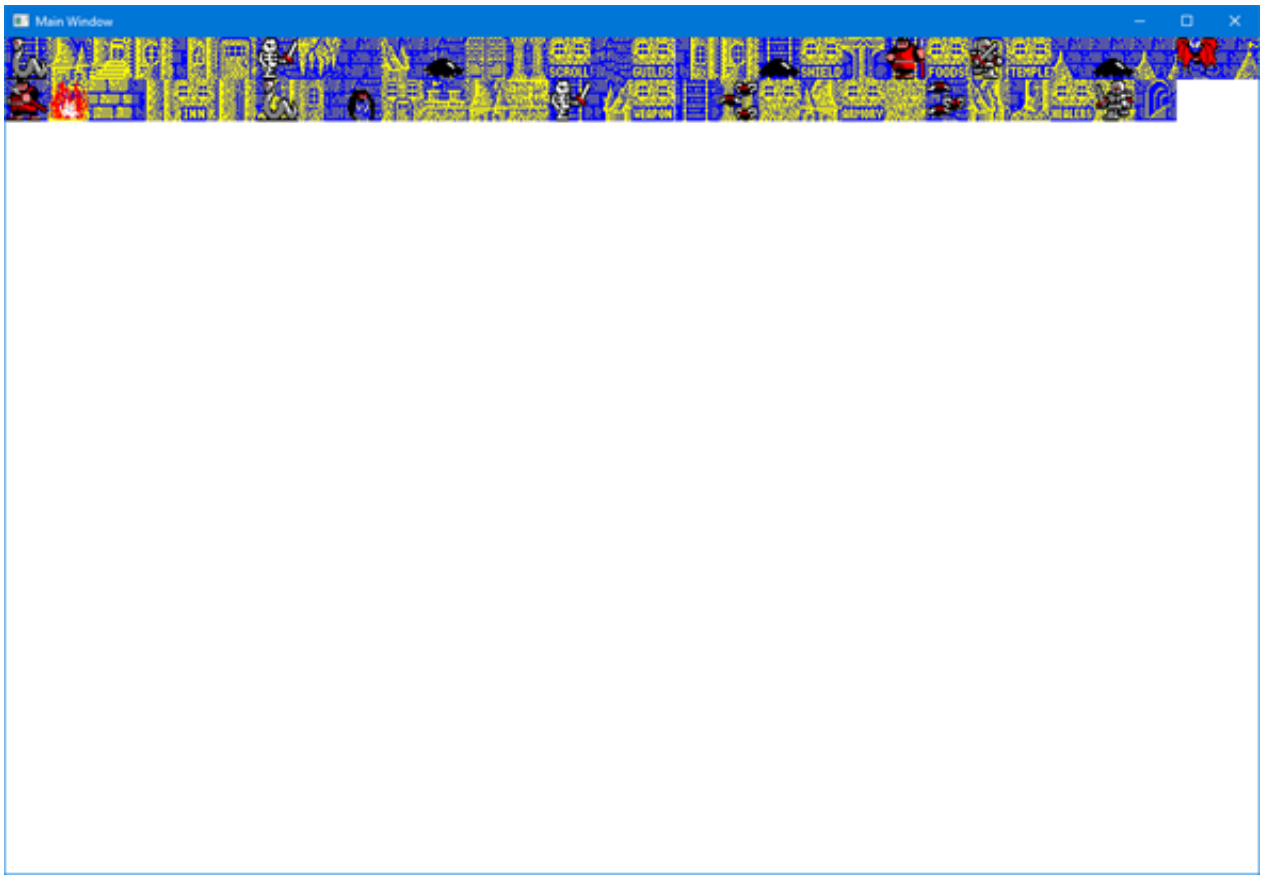

Fig. 2: ps4_2 Output Example

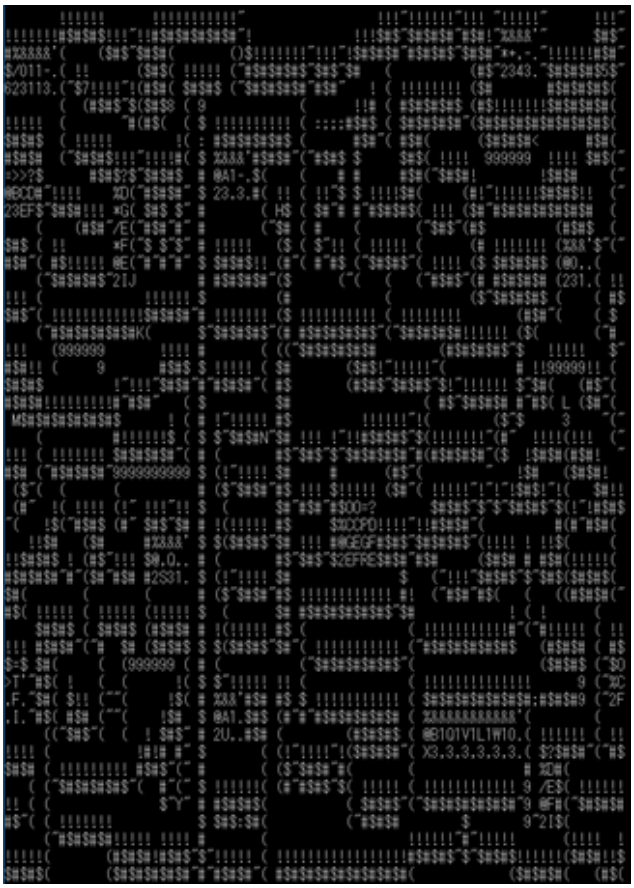

Fig. 3: ps4_2 Output Example

6 Test Your Code on the Compiler Server

Test your source files (.cpp and .h files) on the compiler server. Some assignment may not require .h files. You do not have to test files that you don't make modifications. The files you need to test are the ones you write or modify.

We have four compiler servers:

- http://freefood1.andrew.cmu.edu:24780
- http://freefood2.andrew.cmu.edu:24780
- http://freefood3.andrew.cmu.edu:24780
- http://freefood4.andrew.cmu.edu:24780

Make sure you don't see red lines when you select your files and hit "Compile Test" button on the server.

We have multiple servers to make it less likely that all of them need to shut down for maintenance. If do not have to test on all of the servers. You need to make sure that your code passes on one of the servers.

7 Submit

Lastly, you need to submit using git. What you need to do are two things: (1) add files to git's control, and then (2) send to the git server.

7.1 Add Files to git's control

In this case, you want to add all the files under ps1 subdirectory. To do so, type:

```
git add ~/24783/yourAndrewID/ps4
```

This command will add ps4 directory and all files under the subdirectories.

7.2 Send to the Git Server

In Git, sending files to the server is a two-step process. The first step is local commit. You can do it by:

```
git commit -m "Problem Set 4 solution"
```

The message can be anything, but it is recommended to type something meaningful, at least you can see what changes you made to your repository.

Local commit is just local. Git server does not know about any local commit unless the commit is sent (or pushed) to the server. To do so, type:

```
git push
```

Make sure to do it in the CMU network. If you are working from home (probably most likely), use VPN to connect to the CMU network.

You can re-submit (commit and push) your solution as many times as you want with no penalty before the submission due.

8 Verification

It is recommended to clone your repository to a different location and make sure that all of your files have been sent to the Git server.

You can do the following:

```
cd ~
mkdir 24783Verify
cd 24783Verify
git clone https://yourAndrewID@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

Once you made sure all the files have been submitted, you can delete files and directories under 24783Verify directory.

9 Bonus Problem 1 (+2 points)

In ps4_2, after showing the tiles of the unique blocks, when the user presses the space key, draw the map on the window. However, since the map is large, it does not fit within the window. Therefore, the user must be able to scroll the map by arrow keys. (40 pixels at a time).

Do it without storing the original bitmap. Since you have already assigned a unique ID starting with 0, you can do it by adding the following two data structures.

- A reverse map from a unique ID to a bitmap tile. It can be a map or just an array of bitmaps.
- A 2d map of unique IDs.

With this, you can do it with substantially smaller memory usage than keeping the entire of the original bitmap.

Since this is a bonus problem, TAs will give only limited hints for your questions.

10 Bonus Problem 2 (+3 points on top of Bonus Problem 1)

This requires the previous BONUS PROBLEM 1.

Surprisingly, the background map is using only Blue and Yellow. Other game characters use Red, White, and Black. Therefore, this game only uses five colors in total. It is an art of pixel graphics.

Now, because we know that the map consists only of Blue and Yellow, we can identify background block types where a character is on. (Every character move 40 pixels at a time, by the way).

You can compare a block that includes Red, White, or Black pixel with the matching block with only Blue and Yellow. You can calculate the similarity by comparing how many Blue and Yellow pixels match (ignoring all Red, White, and Black pixels)

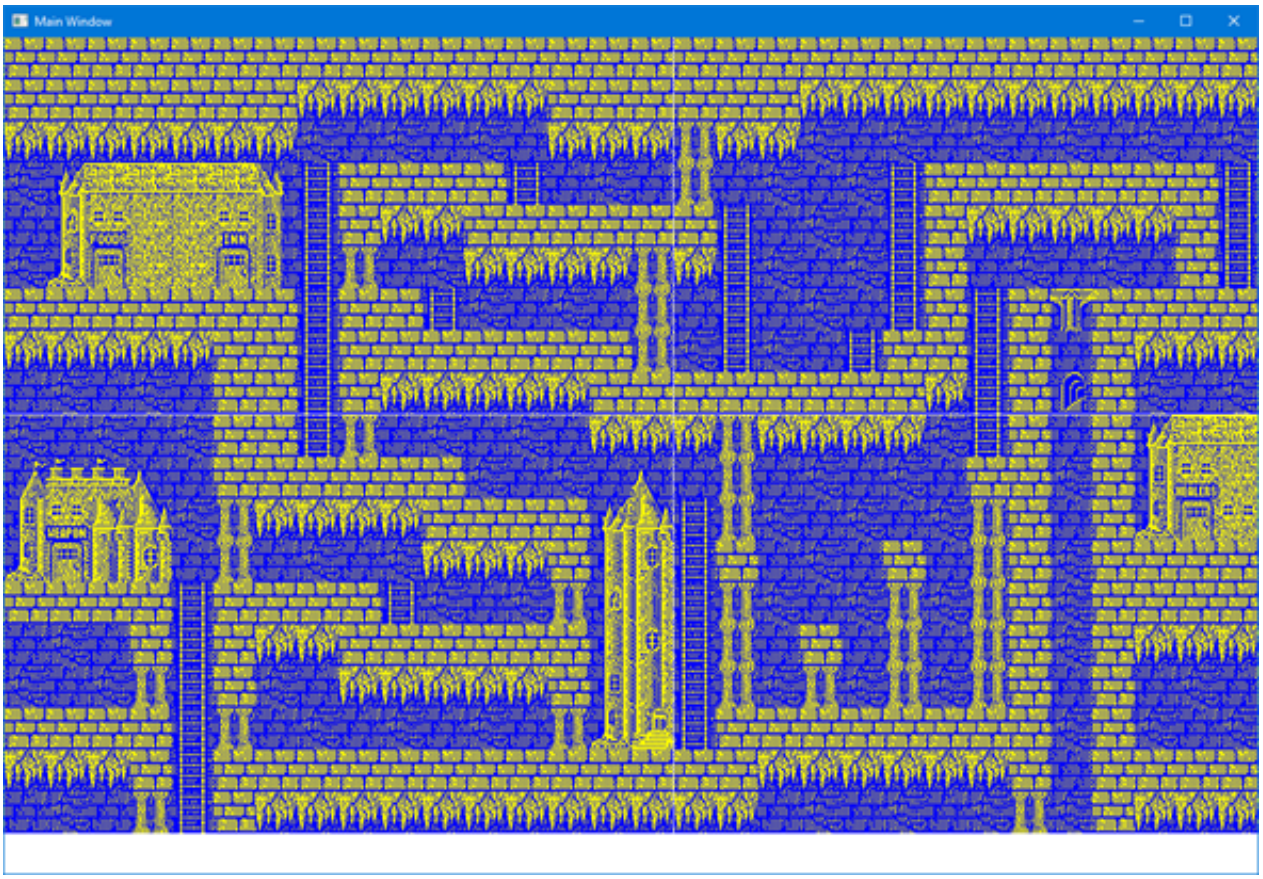

Fig. 4: ps4_2 Bonus Problem 2 Output Example

After satisfying the condition in ps4_2 and the first BONUS PROBLEM, when the user presses the SPACE key for the second time, draw the map on the window, but all characters erased.

If you successfully implement it, you will get an image like Fig. 4 from Level1.png.

11 Bonus Problem 3 (+5 points, independent of Bonus Problems 1 and 2)

Find a music score in png sub-directory of ps4. It is a music score of the BGM of the original Xanadu. Play it as BGM of ps4-2. In the original game, it repeats endlessly, but you don't have to repeat since I haven't finished and tested repeat function of MML Player library yet.

If you go for this bonus problem, in target_link_libraries for ps4_2, add yssimplesound and yssimpldsound_platform at the end. The line should look like:

```
target_link_libraries(ps4_2 fssimplewindow simplebitmap hashutil mmlplayer yssimplesound yssimplesound_platform)
```

If you do all the bonus problems successfully, you can get 10 extra points from this assignment.

Good Luck!