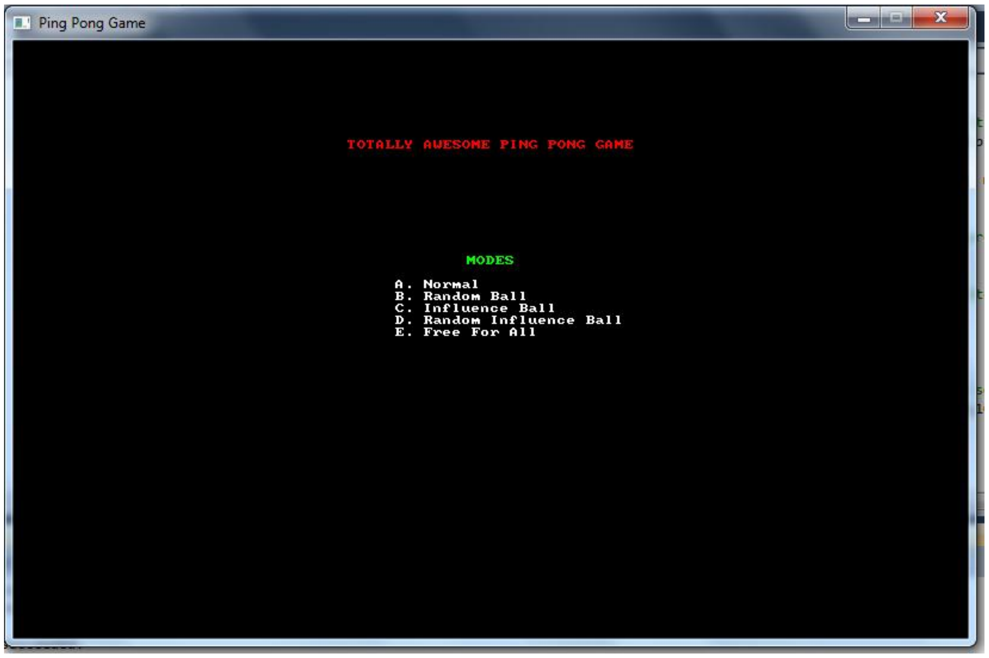
Rich Ung, 003-813-920
Simon Yu, 303-790-516



Ping Pong Game

TOTALLY AWESOME PING PONG GAME

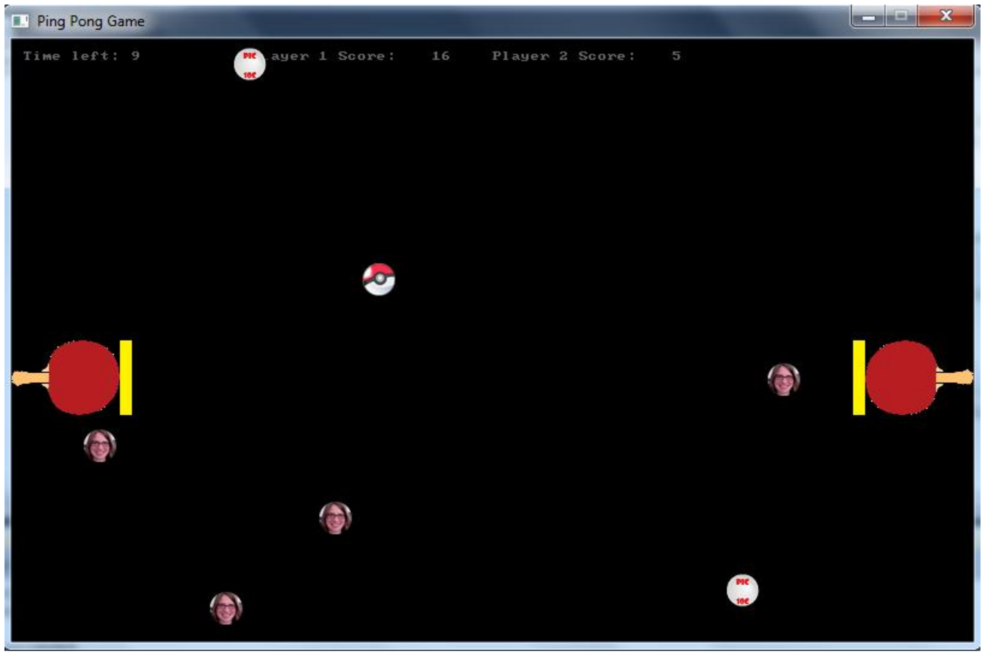
MODES

```
A. Normal
B. Random Ball
C. Influence Ball
D. Random Influence Ball
E. Free For All
```

# Totally Awesome Ping Pong Game Application

**PIC10C Final Project**
**June 13, 2012**

**Rich Ung, 003-813-920, ungrich@gmail.com**
**Simon Yu, 303-790-516, yusimon127@gmail.com**



Ping Pong Game

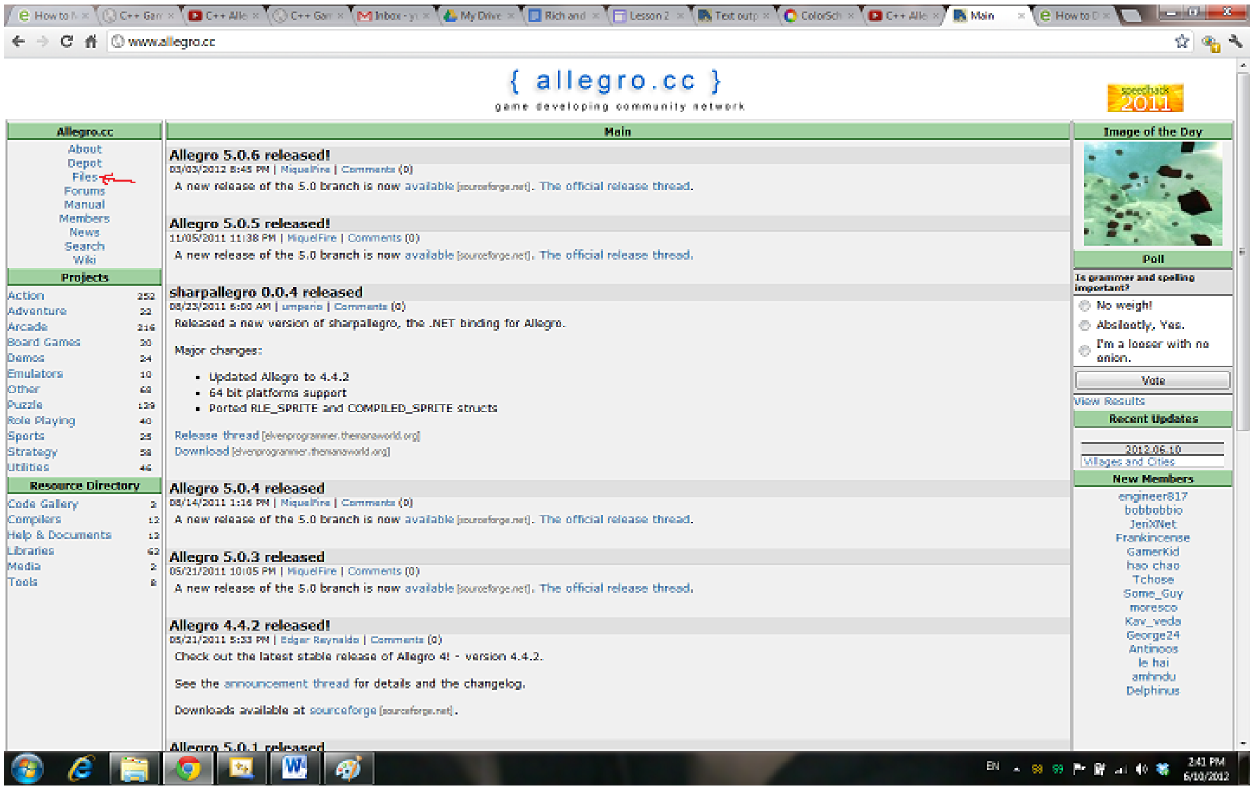
```
Time left: 9        ayer 1 Score:    16     Player 2 Score:    5
```

Rich Ung, 003-813-920
Simon Yu, 303-790-516

# How to install Allegro 4.2 onto Microsoft Visual Studio 2010

1) Go to http://www.allegro.cc

2) Click on Files



**Allegro.cc**
About
Depot
Files
Forums
Manual
Members
News
Search
Wiki

**Projects**

| | |
|---|---|
| Action | 252 |
| Adventure | 22 |
| Arcade | 216 |
| Board Games | 30 |
| Demos | 24 |
| Emulators | 10 |
| Other | 68 |
| Puzzle | 139 |
| Role Playing | 40 |
| Sports | 25 |
| Strategy | 58 |
| Utilities | 46 |

**Resource Directory**

| | |
|---|---|
| Code Gallery | 3 |
| Compilers | 13 |
| Help & Documents | 13 |
| Libraries | 62 |
| Media | 2 |
| Tools | 8 |

**Main**

**Allegro 5.0.6 released!**
03/03/2012 8:45 PM | MiquelFire | Comments (0)
A new release of the 5.0 branch is now available [sourceforge.net]. The official release thread.

**Allegro 5.0.5 released!**
11/05/2011 11:38 PM | MiquelFire | Comments (0)
A new release of the 5.0 branch is now available [sourceforge.net]. The official release thread.

**sharpallegro 0.0.4 released**
08/23/2011 6:00 AM | umperio | Comments (0)
Released a new version of sharpallegro, the .NET binding for Allegro.

Major changes:

- Updated Allegro to 4.4.2
- 64 bit platforms support
- Ported RLE_SPRITE and COMPILED_SPRITE structs

Release thread [elvenprogrammer.theramaworld.org]
Download [elvenprogrammer.theramaworld.org]

**Allegro 5.0.4 released**
08/14/2011 1:16 PM | MiquelFire | Comments (0)
A new release of the 5.0 branch is now available [sourceforge.net]. The official release thread.

**Allegro 5.0.3 released**
05/21/2011 10:05 PM | MiquelFire | Comments (0)
A new release of the 5.0 branch is now available [sourceforge.net]. The official release thread.

**Allegro 4.4.2 released!**
05/21/2011 5:33 PM | Edgar Reynaldo | Comments (0)
Check out the latest stable release of Allegro 4! - version 4.4.2.

See the announcement thread for details and the changelog.

Downloads available at sourceforge [sourceforge.net].

**Allegro 5.0.1 released**

**Image of the Day**

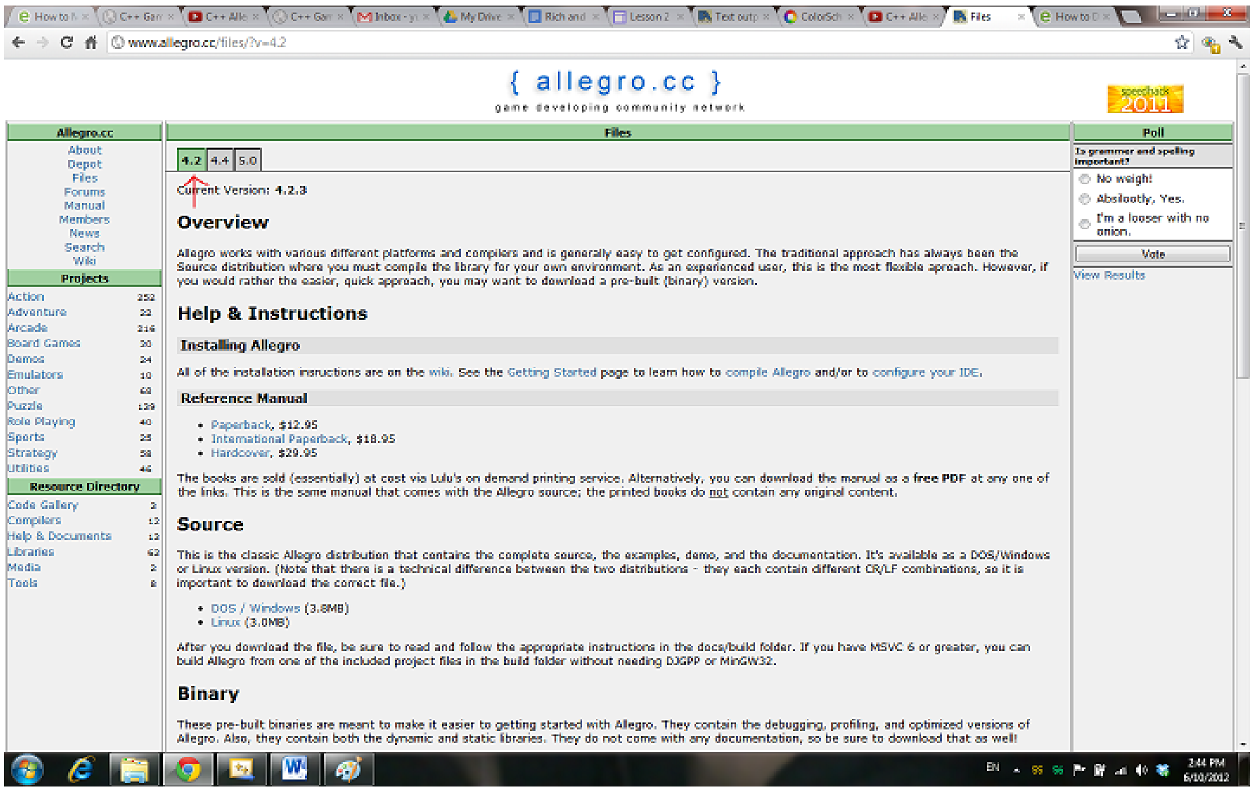
**Poll**
Is grammer and spelling important?
No weigh!
Abslootly, Yes.
I'm a looser with no onion.
Vote
View Results

**Recent Updates**
2012.06.10
Villages and Cities

**New Members**
engineer817
bobbobbio
JeriXNet
Frankincense
GamerKid
hao chao
Tchose
Some_Guy
moresco
Kav_veda
George24
Antinoos
le hai
amhndu
Delphinus

Rich Ung, 003-813-920
Simon Yu, 303-790-516

3) Click on the 4.2 version of Allegro


www.allegro.cc/files/?v=4.2

{ allegro.cc }
game developing community network

| Allegro.cc |
|---|
| About |
| Depot |
| Files |
| Forums |
| Manual |
| Members |
| News |
| Search |
| Wiki |

| Projects | |
|---|---|
| Action | 252 |
| Adventure | 22 |
| Arcade | 216 |
| Board Games | 30 |
| Demos | 24 |
| Emulators | 10 |
| Other | 68 |
| Puzzle | 139 |
| Role Playing | 40 |
| Sports | 25 |
| Strategy | 58 |
| Utilities | 46 |

| Resource Directory | |
|---|---|
| Code Gallery | 3 |
| Compilers | 12 |
| Help & Documents | 12 |
| Libraries | 62 |
| Media | 2 |
| Tools | 8 |

Files

4.2 4.4 5.0

Current Version: 4.2.3

Overview

Allegro works with various different platforms and compilers and is generally easy to get configured. The traditional approach has always been the Source distribution where you must compile the library for your own environment. As an experienced user, this is the most flexible aproach. However, if you would rather the easier, quick approach, you may want to download a pre-built (binary) version.

Help & Instructions

Installing Allegro

All of the installation insructions are on the wiki. See the Getting Started page to learn how to compile Allegro and/or to configure your IDE.

Reference Manual

- Paperback, $12.95
- International Paperback, $18.95
- Hardcover, $29.95

The books are sold (essentially) at cost via Lulu's on demand printing service. Alternatively, you can download the manual as a **free PDF** at any one of the links. This is the same manual that comes with the Allegro source; the printed books do not contain any original content.

Source

This is the classic Allegro distribution that contains the complete source, the examples, demo, and the documentation. It's available as a DOS/Windows or Linux version. (Note that there is a technical difference between the two distributions - they each contain different CR/LF combinations, so it is important to download the correct file.)

- DOS / Windows (3.8MB)
- Linux (3.0MB)

After you download the file, be sure to read and follow the appropriate instructions in the docs/build folder. If you have MSVC 6 or greater, you can build Allegro from one of the included project files in the build folder without needing DJGPP or MinGW32.

Binary

These pre-built binaries are meant to make it easier to getting started with Allegro. They contain the debugging, profiling, and optimized versions of Allegro. Also, they contain both the dynamic and static libraries. They do not come with any documentation, so be sure to download that as well!

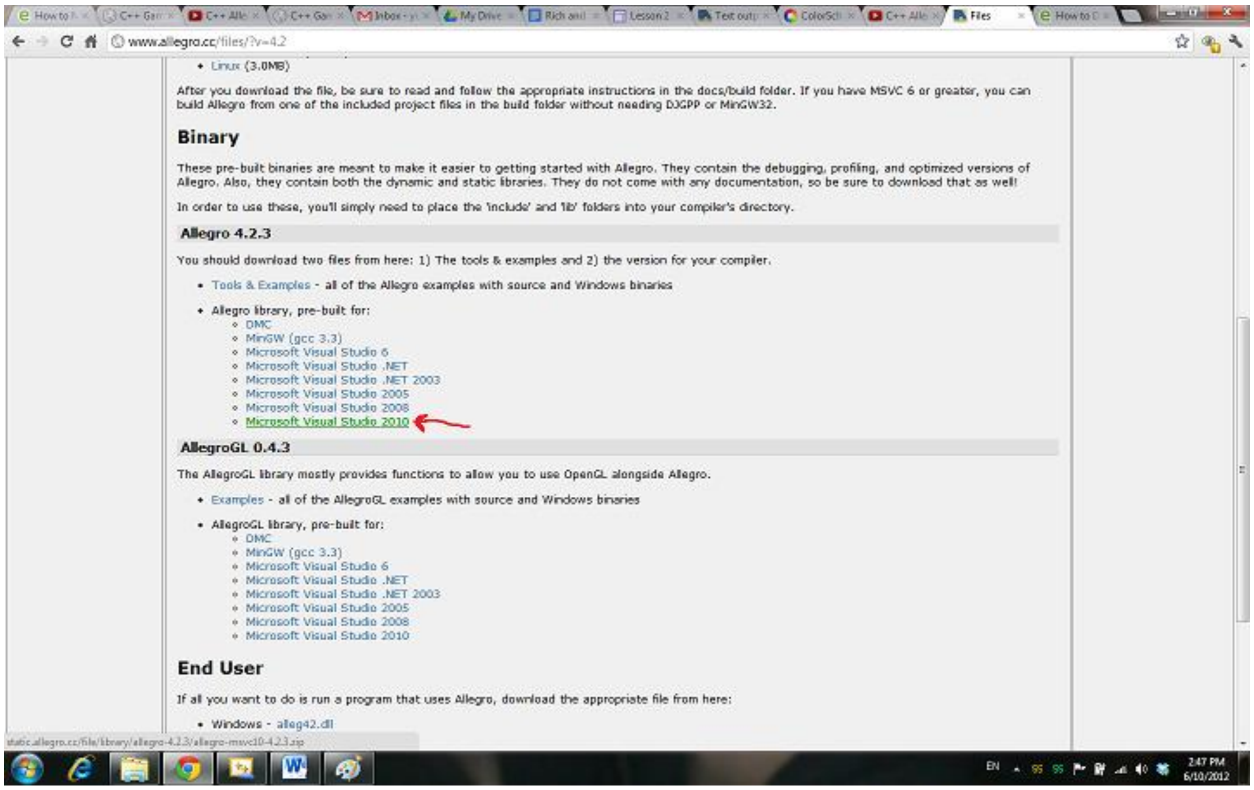
Poll

Is grammer and spelling important?

- No weigh!
- Absilootly, Yes.
- I'm a looser with no onion.

Rich Ung, 003-813-920
Simon Yu, 303-790-516

4) Scroll down to the Binary section and under Allegro 4.2.3, download the version for Microsoft Visual Studio 2010


www.allegro.cc/files/?v=4.2
static.allegro.cc/file/library/allegro-4.2.3/allegro-msvc10-4.2.3.zip

- Linux (3.0MB)

After you download the file, be sure to read and follow the appropriate instructions in the docs/build folder. If you have MSVC 6 or greater, you can build Allegro from one of the included project files in the build folder without needing DJGPP or MinGW32.

**Binary**

These pre-built binaries are meant to make it easier to getting started with Allegro. They contain the debugging, profiling, and optimized versions of Allegro. Also, they contain both the dynamic and static libraries. They do not come with any documentation, so be sure to download that as well!

In order to use these, you'll simply need to place the 'include' and 'lib' folders into your compiler's directory.

**Allegro 4.2.3**

You should download two files from here: 1) The tools & examples and 2) the version for your compiler.

- Tools & Examples - all of the Allegro examples with source and Windows binaries
- Allegro library, pre-built for:
  - DMC
  - MinGW (gcc 3.3)
  - Microsoft Visual Studio 6
  - Microsoft Visual Studio .NET
  - Microsoft Visual Studio .NET 2003
  - Microsoft Visual Studio 2005
  - Microsoft Visual Studio 2008
  - Microsoft Visual Studio 2010

**AllegroGL 0.4.3**

The AllegroGL library mostly provides functions to allow you to use OpenGL alongside Allegro.

- Examples - all of the AllegroGL examples with source and Windows binaries
- AllegroGL library, pre-built for:
  - DMC
  - MinGW (gcc 3.3)
  - Microsoft Visual Studio 6
  - Microsoft Visual Studio .NET
  - Microsoft Visual Studio .NET 2003
  - Microsoft Visual Studio 2005
  - Microsoft Visual Studio 2008
  - Microsoft Visual Studio 2010

**End User**

If all you want to do is run a program that uses Allegro, download the appropriate file from here:

- Windows - alleg42.dll

5) Use WinRAR to open the file allegro-msvc10-4.2.3.zip

6) Double click on allegro-msvc10-4.2.3

7) Open up the folder where you installed Microsoft Visual Studio 2010

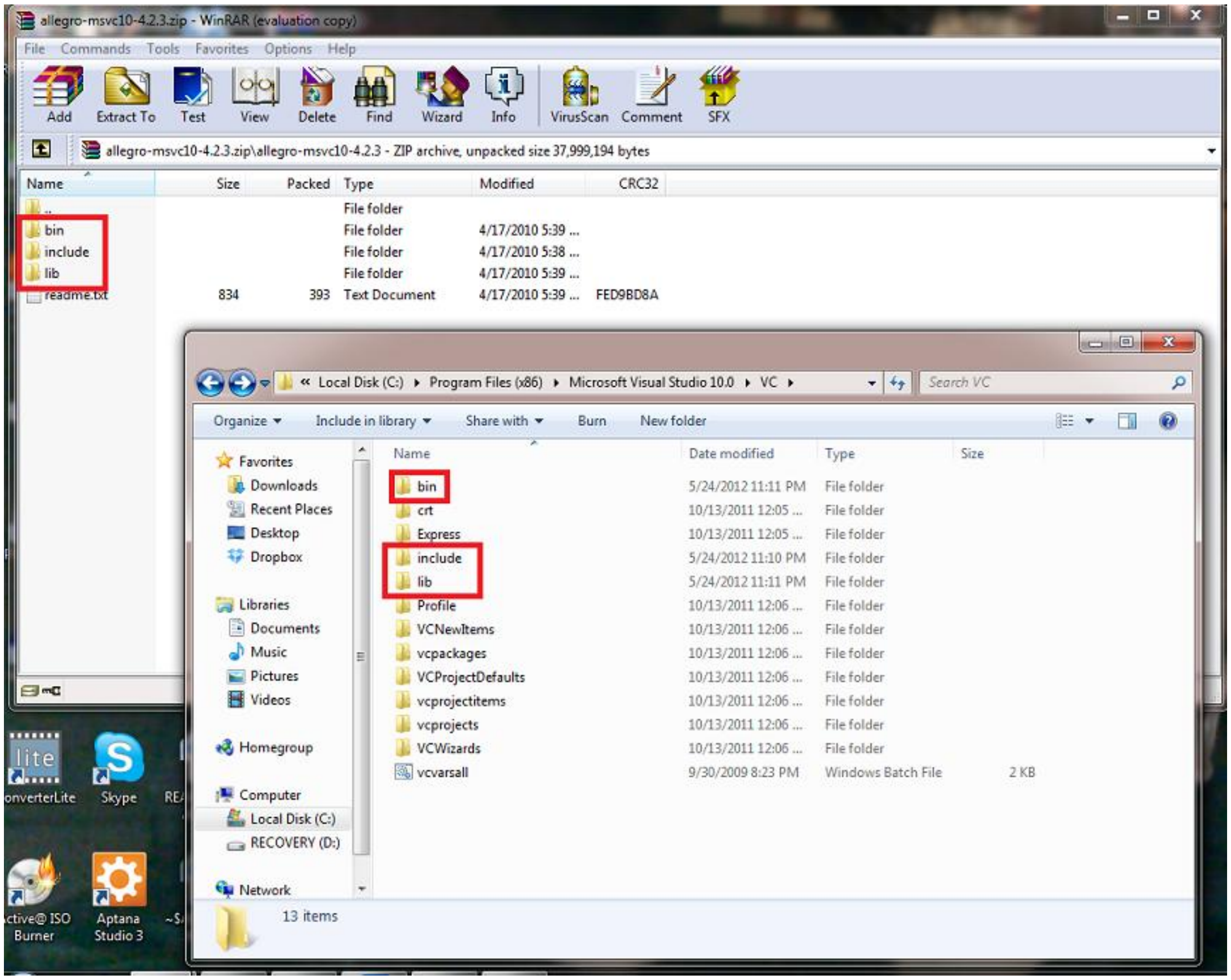
If on windows most likely C:\ Program Files (x86)\ Microsoft Visual Studio 10.0\ VC

Rich Ung, 003-813-920
Simon Yu, 303-790-516

8) Copy the files inside the bin folder of the .zip file into the bin folder of the VC folder

Copy the files inside the include folder of the .zip file into the include folder of the VC folder

Copy the files inside the lib folder of the .zip file into the lib folder of the VC folder


allegro-msvc10-4.2.3.zip - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan Comment SFX
Organize Include in library Share with Burn New folder
Favorites Downloads Recent Places Desktop Dropbox
Libraries Documents Music Pictures Videos
Homegroup
Computer Local Disk (C:) RECOVERY (D:)
Network
13 items
converterLite Skype Active@ ISO Burner Aptana Studio 3

allegro-msvc10-4.2.3.zip\allegro-msvc10-4.2.3 - ZIP archive, unpacked size 37,999,194 bytes

| Name | Size | Packed | Type | Modified | CRC32 |
|---|---|---|---|---|---|
| .. | | | File folder | | |
| bin | | | File folder | 4/17/2010 5:39 ... | |
| include | | | File folder | 4/17/2010 5:38 ... | |
| lib | | | File folder | 4/17/2010 5:39 ... | |
| readme.txt | 834 | 393 | Text Document | 4/17/2010 5:39 ... | FED9BD8A |

« Local Disk (C:) ▸ Program Files (x86) ▸ Microsoft Visual Studio 10.0 ▸ VC ▸

| Name | Date modified | Type | Size |
|---|---|---|---|
| bin | 5/24/2012 11:11 PM | File folder | |
| crt | 10/13/2011 12:05 ... | File folder | |
| Express | 10/13/2011 12:05 ... | File folder | |
| include | 5/24/2012 11:10 PM | File folder | |
| lib | 5/24/2012 11:11 PM | File folder | |
| Profile | 10/13/2011 12:06 ... | File folder | |
| VCNewItems | 10/13/2011 12:06 ... | File folder | |
| vcpackages | 10/13/2011 12:06 ... | File folder | |
| VCProjectDefaults | 10/13/2011 12:06 ... | File folder | |
| vcprojectitems | 10/13/2011 12:06 ... | File folder | |
| vcprojects | 10/13/2011 12:06 ... | File folder | |
| VCWizards | 10/13/2011 12:06 ... | File folder | |
| vcvarsall | 9/30/2009 8:23 PM | Windows Batch File | 2 KB |

Rich Ung, 003-813-920
Simon Yu, 303-790-516

9) Open up My Computer and type in system32 into the search box. Open up your C:\system32 folder

10) Copy the files from the bin folder from the .zip file into the system32 folder

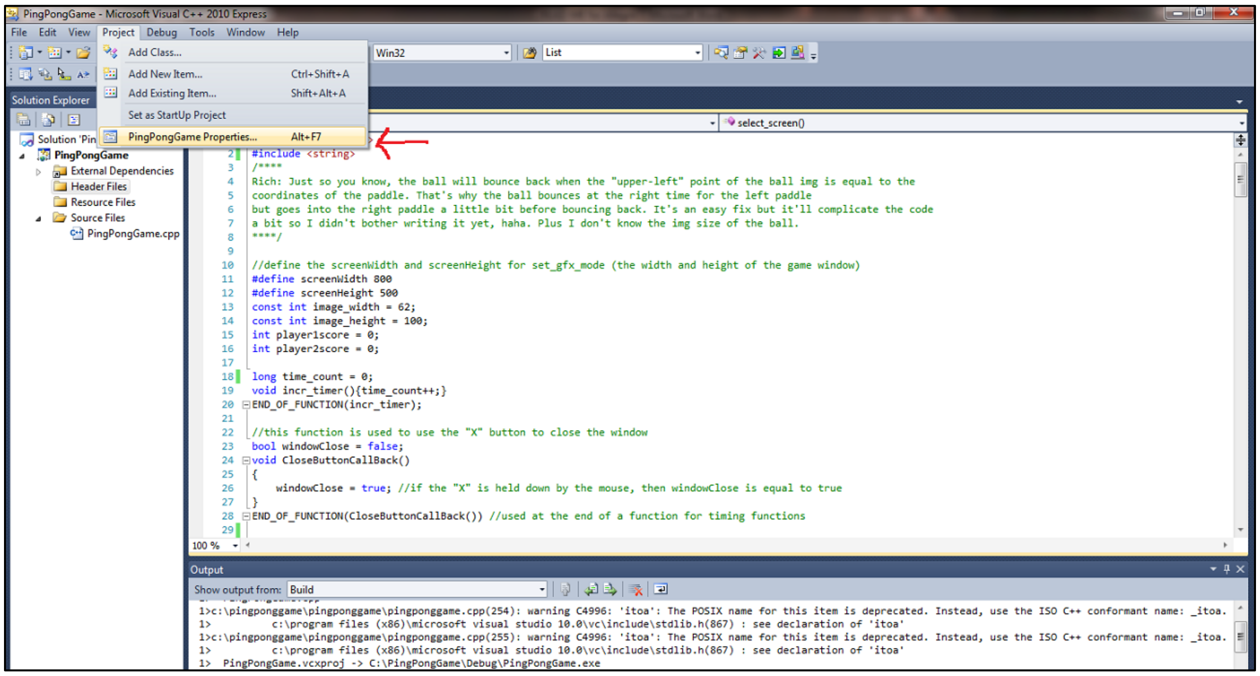
11) Open up Microsoft Visual Studio 2010 and create a new project.

12) Make sure it is a Win32 project. Click next and make sure "Empty Project" is checked. Then click Finish.

13) Add existing header (.h) files that we turned in into the "Header Files" folder.

14) Add existing .cpp files that we turned in into the "Source Files" folder.

15) We now have to change the project properties. Go to Project -> *ProjectName* Properties



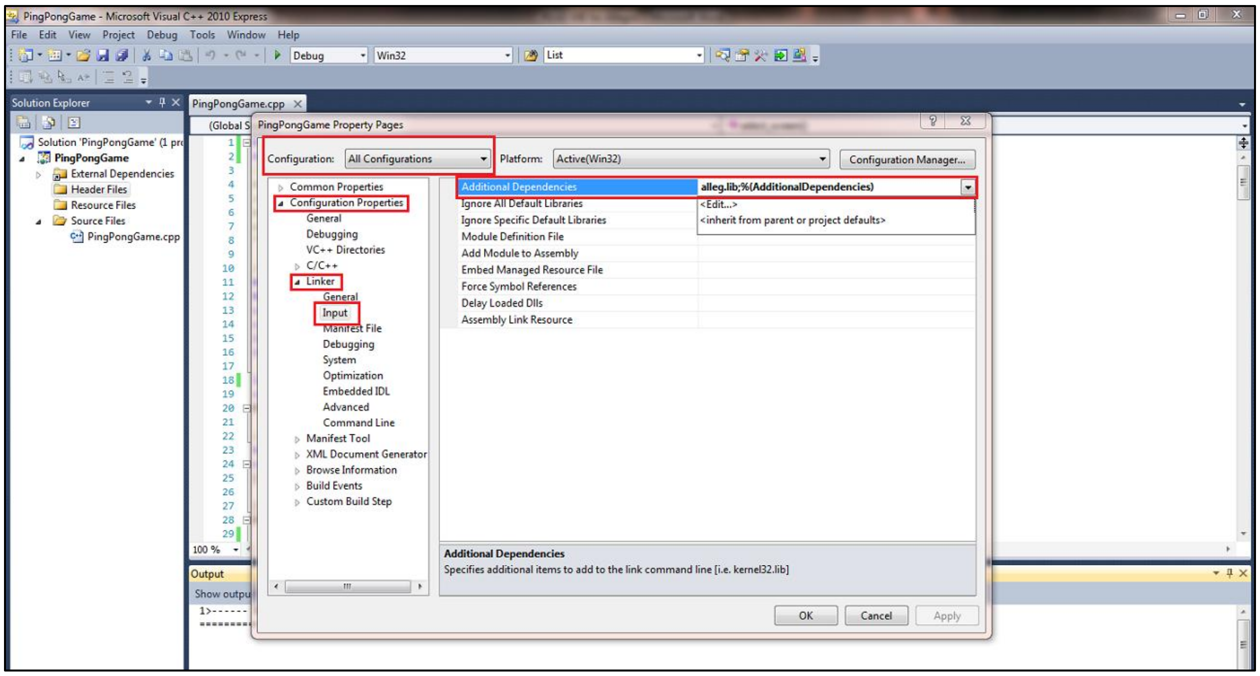
```
#include <string>
/****
Rich: Just so you know, the ball will bounce back when the "upper-left" point of the ball img is equal to the
coordinates of the paddle. That's why the ball bounces at the right time for the left paddle
but goes into the right paddle a little bit before bouncing back. It's an easy fix but it'll complicate the code
a bit so I didn't bother writing it yet, haha. Plus I don't know the img size of the ball.
****/

//define the screenWidth and screenHeight for set_gfx_mode (the width and height of the game window)
#define screenWidth 800
#define screenHeight 500
const int image_width = 62;
const int image_height = 100;
int player1score = 0;
int player2score = 0;

long time_count = 0;
void incr_timer(){time_count++;}
END_OF_FUNCTION(incr_timer);

//this function is used to use the "X" button to close the window
bool windowClose = false;
void CloseButtonCallBack()
{
    windowClose = true; //if the "X" is held down by the mouse, then windowClose is equal to true
}
END_OF_FUNCTION(CloseButtonCallBack()) //used at the end of a function for timing functions

```

```
1>c:\pingponggame\pingponggame\pingponggame.cpp(254): warning C4996: 'itoa': The POSIX name for this item is deprecated. Instead, use the ISO C++ conformant name: _itoa.
1>          c:\program files (x86)\microsoft visual studio 10.0\vc\include\stdlib.h(867) : see declaration of 'itoa'
1>c:\pingponggame\pingponggame\pingponggame.cpp(255): warning C4996: 'itoa': The POSIX name for this item is deprecated. Instead, use the ISO C++ conformant name: _itoa.
1>          c:\program files (x86)\microsoft visual studio 10.0\vc\include\stdlib.h(867) : see declaration of 'itoa'
1>  PingPongGame.vcxproj -> C:\PingPongGame\Debug\PingPongGame.exe
```

Rich Ung, 003-813-920
Simon Yu, 303-790-516

16) Under Configurations, go to “All Configurations”

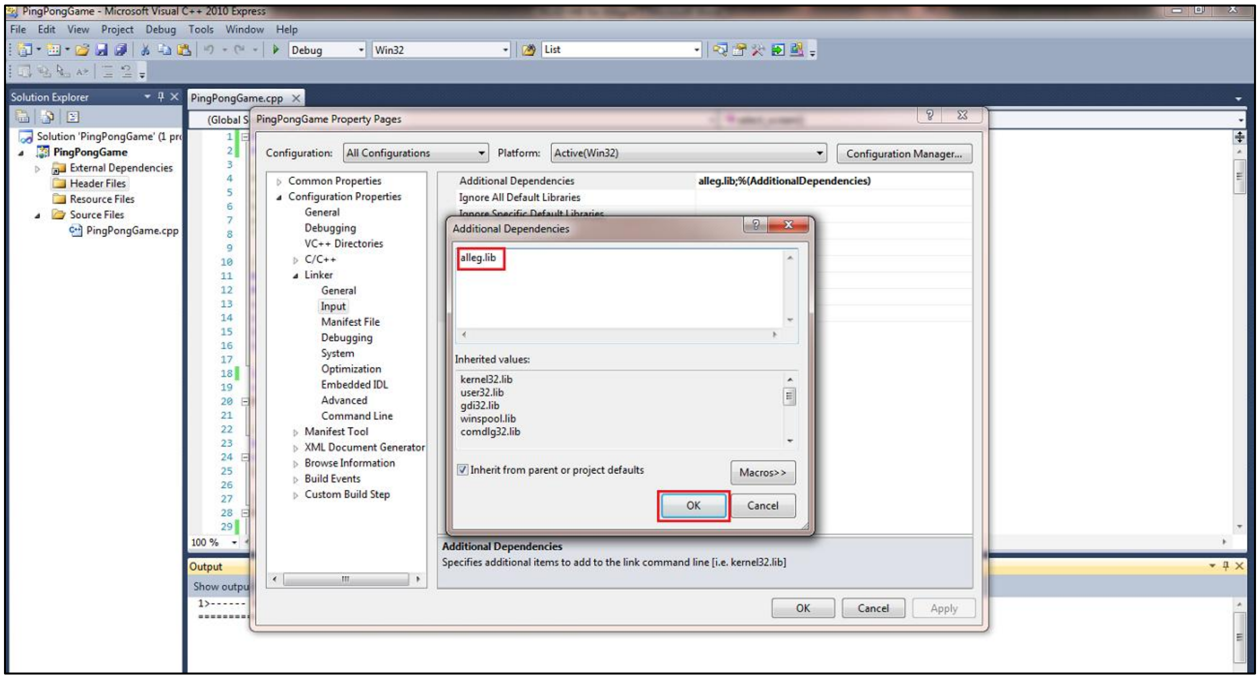
17) Under Configuration Properties -> Linker -> Input, change the Additional Dependencies by clicking it, then Edit… , and then type in alleg.lib, then click Ok



Configuration: All Configurations
Platform: Active(Win32)

| | |
|---|---|
| Additional Dependencies | alleg.lib;%(AdditionalDependencies) |
| Ignore All Default Libraries | <Edit...> |
| Ignore Specific Default Libraries | <inherit from parent or project defaults> |
| Module Definition File | |
| Add Module to Assembly | |
| Embed Managed Resource File | |
| Force Symbol References | |
| Delay Loaded Dlls | |
| Assembly Link Resource | |

**Additional Dependencies**
Specifies additional items to add to the link command line [i.e. kernel32.lib]


PingPongGame - Microsoft Visual C++ 2010 Express
PingPongGame Property Pages
Additional Dependencies
Macros>>
OK
Cancel

Configuration: All Configurations
Platform: Active(Win32)

Additional Dependencies: alleg.lib;%(AdditionalDependencies)

Additional Dependencies

alleg.lib

Inherited values:

kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib

☑ Inherit from parent or project defaults

**Additional Dependencies**
Specifies additional items to add to the link command line [i.e. kernel32.lib]

Rich Ung, 003-813-920
Simon Yu, 303-790-516

## **NOTE ABOUT RUNNING OUR PROGRAM**

- Make sure to include the image files (.jpg, .bmp, etc) in the same folder as the .h and .cpp files
- In Visual Studio:
    - Put .h files in “header files” of Visual Studio program
    - Put .cpp files in “source files” of Visual Studio program
- The allegro installation .zip file (allegro-msvc10-4.2.3.zip) is included in the .zip file, so you don’t need to download it

The program uses **Allegro 4.2**, not Allegro 5!

Rich Ung, 003-813-920
Simon Yu, 303-790-516

## Allegro 4.2 Key Terms/Functions

allegro_init

- Initializes the allegro environment so that we can use their functions

install_keyboard

- Sets up the system so we can do user key input

install_timer()

- Sets up the timer routine used for the game loop

set_color_depth(32)

- Sets the global pixel color depth

set_gfx_mode(…)

- Sets the game window size

set_window_title(…)

- Sets the title of the game at the very top

install_int_ex(incr_timer, BPS_TO_TIMER(60))

- used to start the timer routine at 60 frames per second

set_close_button_callback(CloseButtonCallBack)

- sets up the close "X" button so that when the user clicks "X", the program will exit and close

allegro_exit()

- so we completely close and delete everything from the program at the end of main

END_OF_MAIN

- to enable cross-platform compatibility

Rich Ung, 003-813-920
Simon Yu, 303-790-516

END_OF_FUNCTION, LOCK_FUNCTION, LOCK_VARIABLE

- They all holds the place in memory and prevent it from moving somewhere else when some memory are swapped (avoiding interrupts)

key[KEY_A], key[KEY_B], key[KEY_C], etc

- Allows the user to input a keyboard key into the keyboard buffer

keypressed()

- Returns true if there are keyboard keys waiting in the keyboard buffer

readkey()

- Returns the next keyboard key waiting in the keyboard buffer

clear_keybuf(…)

- Clears the keyboard buffer

textout_centre_ex(…), textout_ex(…)

- Allows the user to output text to the screen

makecol(…)

- A function that make a color, takes in R, G, B colors

create_bitmap(…)

- Creates a sheet that allows programmer to draw and output text on it, load image onto it

draw_sprite(…)

- Draws an image onto the BITMAP

load_bitmap(…)

- Loads an image to the BITMAP

screen

- The main BITMAP screen after one run-through of the game loop

Rich Ung, 003-813-920
Simon Yu, 303-790-516

BITMAP* Buffer

- A Buffer Is used to hold a BITMAP after the game loop ends and loops back to the beginning. If only screen is used, it will be a tiny delay before the screen loads up again. So we need the Buffer to prevent flickering.

blit(…)

- Merges two screens together (used for merging Buffer and screen to prevent flickering

clear_bitmap(...)

- Clears the whole bitmap to a color depth of 0 (which is a black screen)

Rest(…)

- Pauses the program for a certain number of milliseconds

Rich Ung, 003-813-920
Simon Yu, 303-790-516

## Gameplay Instructions

**Selection Screen**

Press the corresponding key to play the corresponding mode:

**A:** Normal (A normal game of ping pong)

**B:** RandomBall (The ball moves in a random velocity and acceleration everytime it hits a boundary)

**C:** InfluenceBall (By pressing the A and D keys for player 1 or the Left-Arrow and Right-Arrow keys for player 2, you can easily make the ball on the field go faster or slower to harm your opponent)

**D:** RandomInfluenceBall (This ball has both characteristics of the RandomBall and InfluenceBall, it moves in a random velocity and acceleration when it hits a boundary and you may use the A/D and Right/Left keys to make the ball move faster/slower)

**E:** Free for all (Press the E key to release a RandomBall, the R key to release a InfluenceBall, the T key to release a RandomInfluenceBall, and you may use the A/D and Right/Left keys to make the ball move faster/slower)

**High Score Screen**

If a player wins, and a score is high enough, you can enter high score screen.

-Enter name.

-The name is outputted to the file “highscore.txt” along with their score.

-NOTE: Cannot press space, or enter a name more than 10 characters long. We threw an exception. =]

-If the name is empty (if the player doesn’t enter a name and just presses enter, the name does not get outputted to file.

DO NOT ALTER THE OUTPUT FILE DIRECTLY. IT IS MEANT FOR THE PROGRAM ONLY. YOU CAN OPEN IT, BUT PLEASE DO NOT MODIFY ANYTHING.

Rich Ung, 003-813-920
Simon Yu, 303-790-516

## Google Doc Notes

**Introduction**
Instead of a console application, we are going to make a graphics application.

- The graphics application will involve using **Allegro 4.2** to create the program using C++
    - -There will be directions on how to install and run the program in our project
- The classes that might will create:
    - pingPongPaddle (to draw the ping pong paddle)
    - pingPongBall (to draw the ping pong ball)
    - Ball class that allows us to move the ball
    - movePaddle functions to move the paddle
    - screen_selection() function to display screen
- We will allow user input by reading:
    - -Keyboard arrow keys (Up and Down) for Player 1
    - -Keyboard letters (W and S) for Player 2
- By using a time function, we can control the ball's movement as a series of pause frames to give the illusion of continuous movement.

AB's Comments: Looks like a good plan. I like the google doc. How are you planning to divide the work? And you might want to think about starting by using the graphics class from the beginning of our book rather than jumping into the more difficult graphics right away.
Rich: We already looked into the code from the graphics class from the beginning of the book. We thought that we should learn more about how Allegro code works on our own before dividing the work, too.

Rich Ung, 003-813-920
Simon Yu, 303-790-516

## Google Doc Notes (continued)

**Bugs encountered along coding the program**
-All of the common memory errors (they are really easy to get when making a list of pointers that requires iterators)
-Having messages from exceptions show properly
-Making sure there is always a virtual destructor for any base class (otherwise you get an error when deleting a base pointer to a derived class)
-Making sure the ball bounces back and does not phase through the paddle

**Simon's Contribution**
-Constructed main interface of the game
-Made high score output to file & interface
-Made select screen & interface
-Made PDF file on how to install Allegro
-Made Key Terms list to explain Allegro terms

**Rich's Contribution**
-Made Ball Class & Ball subclasses
-Made Inventory Class
    -Can create different types of balls and play with multiple balls at once
-Compiled list of things to cover

**What We Learned**
-Making a graphics program on C++ is harder than it looks. x.x
-Maybe we should have taken the final.
-We used passing by reference to make the paddle move, otherwise it won't move at all because the variables of the ball movement speed doesn't change.
-Communication/Comments between code is important. Unlike the homework, the comments need to be clear so we can actually improve the code and understand what everything does. Otherwise we'd have to call and meet up more often.
-Wasted 2 hours on fixing memory leaks to find out that to always put a virtual destructor on the base class, even if it does nothing.
-Learned about const_iterator to make an iterator declared as const in order to use const functions
-Learned that you can waste hours of debugging by forgetting to put std:: in front of a standard type like (std::string, std::fstream, std::ofstream, std::ifstream)
-Creating a BITMAP* Buffer is useful (to prevent flickering), but more complicated than it seems
-char buffers, BITMAP buffers, keyboard buffers, and other buffers to hold temporary data are important in real world programs
-Use of physics to control how to ball moves
-Learned that programming is actually fun, not just textbook homework problems and stuff
-Learned that games involve a lot of rule thinking
-Be sure to erase and destroy anything that is a pointer, including BITMAP* which is native to the Allegro program (we struggled with memory leaks dealing with the Allgero program itself that resulted in crashes)

-We also witnessed using Task Manager and seeing the Memory (Private Working Set) skyrocket up as the program run, which lead to a crash once it hit 22k per second. (and it took a lot of the CPU useage as well)

**Requirements**
-The final outcome should use all of the sections we have covered. If you decide *not* to use something, you must clarify with me as you go along why you have chosen not to use it.
-The project must have a user's manual (in the form of a readme.txt, or a fancier instruction booklet); this should guide me and any user of any level in how to install necessary software (less than a 10-minute process only) and how to make the project run. It should also include a description of what the project is and what it does. Any project without this, or for which this readme doesn't tell me everything I need to know will receive a C or lower.
-The whole project (all necessary files, the readme, and as much software as is needed) should be included in one folder, and then zipped.
-Each person in the group is to independently document what they personally have contributed to the project, as well as what their group members have contributed. I also want a list of major bugs along the way (even the silly ones) and what you have learned in the process.

Rich Ung, 003-813-920
Simon Yu, 303-790-516

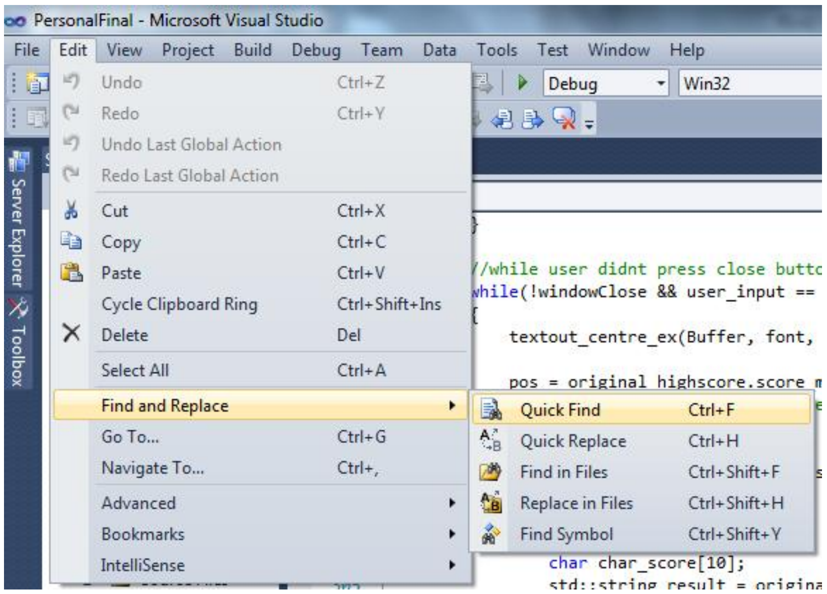
# Topics Covered in PIC10C

<u>**How to search our project to see if we covered the topics**</u>

-We allocated every topic covered in class with a bracket reference in bold (i.e. [14.1]) starting on the next page (Source: Last day of class's outline of the course). Every bracket reference is either located in our .h or .cpp files to easily show where we used the concept for our project, or explained below about why we chose to not utilize that particular topic for our final project.

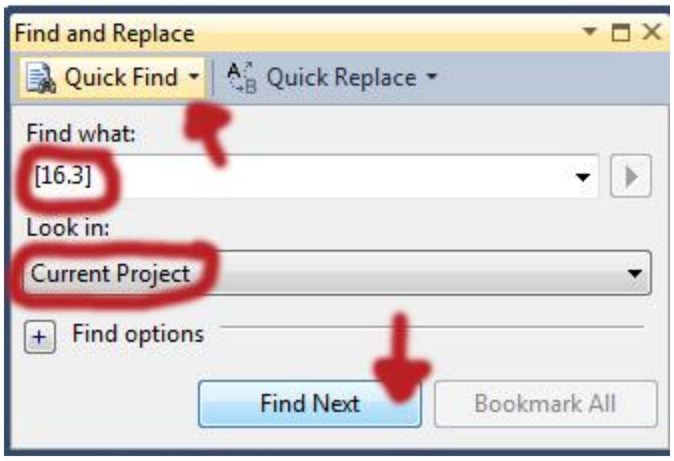
-How to use the bracket reference on Visual Studio 2010:

1) Click on Edit > Find and Replace > Quick Find (or hit Ctrl + F)



```
}
//while user didnt press close buttc
while(!windowClose && user_input ==
{
    textout_centre_ex(Buffer, font,

    pos = original_highscore.score m

        char char_score[10];
        std::string result = origina
```

2) A "Find and Replace" window will pop up. Make sure "Quick Find" is highlighted, enter the bracket reference you want to look for, and make sure "Current Project" is selected for "Look in:" to search through all of the .h and .cpp files within the project. Click on "Find Next".



Find what: [16.3]

Look in: Current Project

Rich Ung, 003-813-920
Simon Yu, 303-790-516

# Topics Covered in PIC10C

## Chapter 14: Operator Overloading

-Overloading

**[14.1]** -Simple Arithmetic
-there was no need to overload the arithmetic operators because it would have made the code harder to read

**[14.2]** -Comparison Operator
-using the .equals function in the highscores.cpp was actually more clear than overloading the comparison (==) operator

**[14.3]** -Input / Output
highscores.cpp (output operator overloaded)

**[14.4]** -Increment and Decrement
-the ++ and -- operators are ambiguous for our functions (it can do different things), for example, the increase and decrease speed functions (2 for each player)

**[14.5]** -Assignment
-inventory.cpp

**[14.6]** -Conversion
-we didn't need to convert any types to another type (none of our classes require conversion to or from another class)

**[14.7]** -Subscript
-inventory.cpp

**[14.8]** -Function Call
-there was no need to call a function like a Matrix, the subscript operator does the job

## Chapter 15: Memory Management

**keep throwing new balls, and when player loses, delete ball

**[15.1]** -Categories of Memory
-Code Memory: the code that we wrote
-Static Data Memory: we used global variables
-Most of our functions are allocated in the stack
-But our balls are dynamically allocated in the heap because they use new/delete.

**[15.2]** -Common Memory Errors
-inventory.cpp (got these errors while coding T.T, stupid list of pointers)
- Initialization errors
  - -we initialized every variable
- Lifetime errors
  - -avoided dangling pointers that points to a location that is no longer valid (make a function that returns a *(pointer) and establishes the address within the function)
  - -avoided returning reference to a local variable (make a function that returns a type& and establishes the variable it returns within the function)

**[15.2.3]** - Array bounds errors
-we made sure that we avoid writing past an array
- Object slicing
-We avoided this problem using typedef and dynamic casting to make sure objects are never sliced.
- Memory leaks
-delete ball objects
-destroy_bitmap(Buffer)
- Using invalid memory references
-we made sure iterators do not access invalid memory such as NULL and others
- Deleting a value more than once
-we set pointers to NULL after they are deleted just in case

**[15.3]** - Constructors
- covered wherever we had classes

**[15.4]** - Destructors
-inventory.cpp

[**15.5]** - Reference Counting
-There was no need to reference count different pointers to the same object. While we could do that for the ball class since we dynamically allocated them, there was no purpose to have more than one pointer to one ball.

# Chapter 16: Templates

**inventory class takes in different types of balls

**[16.1]** - Template Functions
-inventory.h

**[16.2]** - Compile-Time Polymorphism
-inventory.h

**[16.3]** - Template Classes
-None of our classes needed a template. We already had a class that handles all the balls through a list of pointers and that handled different types of classes.
-Note: Optional class “obstacle.h” uses this concept

**[16.4]** - Turning a Class into a Template
-None of our classes needed a template. We already had a class that handles all the balls through a list of pointers and that handled different types of classes.
-Note: Optional class “obstacle.h” uses this concept

**[16.5]** - Non-type Template Parameters (template <typename T, int N>)
-None of our classes needed a template. We already had a class that handles all the balls through a list of pointers and that handled different types of classes.
-Note: Optional class “obstacle.h” uses this concept

**[16.6]** - Setting Behavior Using Template Parameters (ie. passing a function/function object into the template)
-None of our classes needed a template. We already had a class that handles all the balls through a list of pointers and that handled different types of classes.

Rich Ung, 003-813-920
Simon Yu, 303-790-516

NOTE: The multimap class already has one. See pg. 769.
-It was used to automatically sort our high scores.
template< typename K, typename V, typename CMP = less<K> >
class multimap { … }

**Chapter 17: Exception Handling**

**select screen, if user inputs wrong key, throw an exception
**outputting file, if user puts in a name too long (>10 characters?), throw an exception

**[17.1]** - Handling Exceptional Situations
-main.cpp

**[17.2]** - Alternative Mechanisms for Handling Exceptions
-main.cpp

**[17.3]** - Exceptions

**[17.4]** - How to throw them
-main.cpp

**[17.5]** - Exception inheritance hierarchy
-main.cpp

**[17.6]** - Rethrowing exceptions (used when dealing with pointers)
-None of our exceptions required us to delete pointers

**Chapter 18: Name Scope Management**

**[18.1]** - Encapsulation
-classes have variables (class scope)
-brackets have variables (block scope)
-functions have variables (local scope / bracket scope)
-parameters have variables (local scope)
-global variables (global scope)

**[18.2]** - Name scopes
-classes have variables (class scope)
-brackets have variables (block scope)
-functions have variables (local scope / bracket scope)
-parameters have variables (local scope)
-global variables (global scope)

**[18.3]** - Protected scope
-ball.h

**[18.4]** - Friends
-highscores.h

**[18.5]** - Nested Classes
-Because we used private inheritance for the HighScoresIterator, and we needed to access the interator (from outside) the multimap that was initialized in the class, we couldn't set HighScoresIterator as a nested class

**[18.6]** - Private Inheritance (do not want public interface from base class revealed)
-highscores.h

**[18.7]** - Namespaces
-highscores.h

Rich Ung, 003-813-920
Simon Yu, 303-790-516

# Chapter 19: Class Hierarchies

**[19.1]** - Class Inheritance Hierarchies
-ball.cpp
**[19.2]** - Abstract Classes
-ball.cpp
**[19.3]** - Obtaining run-time type information
-inventory.cpp
**[19.4]** - Multiple Inheritance
-ball.cpp
**[19.5]** - Virtual Inheritance
-ball.cpp
**[19.6]** - Software Frameworks
-main.cpp, allegro.h

# Chapter 20: STL

**store high scores into a list (10 scores), iterate through the list to display with an iterator
**[20.1]** - The STL
-highscores.h (used multimap)
-inventory.h
**[20.2]** - Iterators
-inventory.cpp, highscores.cpp
**[20.3]** - Fundamental Containers (vector, list, deque)
-inventory.h
**[20.4]** - Container Adapters (stack, queue, priority queue)
-list and multiset were the only necessary containers we needed for our game
**[20.5]** - Associative Containers (set, multiset, map, multimap)
-highscores.cpp
- (Not covering 20.6 - 20.10)

Rich Ung, 003-813-920
Simon Yu, 303-790-516

**Ping Pong Game (Simon Yu's Journal)**

## Contribution

The ping pong game is a 2-player game that requires keyboard input from both players. Our program uses Allegro (a gaming library). I constructed the main interface of the ping pong game. I tried to make the interface as user-friendly as possible so that even a person that doesn't know how programming works can easily run the program. The program starts off with a selection screen, which I created the interface for as well, that lists the different modes that the user can choose to play. The interface has a layout that is easy to read and understand. If the player presses "A" , then a certain mode will be played. If the player presses "B", then another mode will be played. Once the game starts, the 2 players will either use "w" / "s" to move player 1's paddle up and down or up and down arrow to move player 2's paddle up and down. The functionality and movement of the ball was constructed by Rich. In normal mode, the ball goes in random speeds (it's not at constant speed because we want to add some excitement to a ping pong game). In another mode, the 2 users are allowed to add balls or change ball directions or even make the balls go faster. We also included a Barbaro Ball. The third part of the game was the high score screen interface. I basically made it so that if the player's score is higher than the high scores in the past (which are documented in a text file), then they can enter the high score screen, which displays the past high scores and allows the user to input their name. After the user presses enter, user's name and score is updated to the file. I also made a PDF file about how to install Allegro onto a Window's machine, specifically Microsoft Visual Studio 2010 (Express). Since not many people are familiar with Allegro, I also made a list of key terms that explained what each of the functions that we used do. . I also grasped better understanding of different kinds of buffers are for and how they are used (including keyboard buffer, BITMAP buffer, and char buffers).

## Bugs/Problems along the way

Debugging the code was the most difficult part of programming the interface. Since I never learned Allegro before, dealing with the BITMAP* Buffer was difficult because we had to clear the buffer every time a while loop restarts. I also made a silly mistake, thinking that because we were using Allegro, we can't use #include <fstream>, which was for file input and output. However, because I didn't use using namespace std; I couldn't call the ofstream. I totally forgot to instantiate it as std::ofstream. Hours later, I figured It out and realized I wasted those hours debugging a simple mistake. I also didn't know how to send a function into a different .cpp file (different from main) because the main.cpp code was getting too long. I learned that you have to declare a prototype of that function into a .h file, and then make a .cpp file to write the code for what the function does. I spent an hour or so trying to figure this out. Also, because we are using Allegro, getting user-inputted text to save into a string variable and display to the screen was a little more difficult. Basically we needed a function to convert the key-press as an ASCII number, which we can use if statements to do the functionalities. We also had to use a Buffer to allow it to stay on the screen without flickering. There was also a major bug when I tried to throw a length_error exception, where the message would display forever, or if I try to clear the Buffer after displaying the text, the text will not even display at all. I figured out that I just had to write it to the "screen" instead of "Buffer" 5 hours later.

Rich Ung, 003-813-920
Simon Yu, 303-790-516

**Ping Pong Game (Rich Ung's Journal)**

## My Contribution for the Project

For my contribution for the project, I created, designed, and implemented the Ball abstract class and its three derived classes through multiple inheritances, the Inventory class to keep track of all of the Balls through dynamically allocating memory, and the Obstacle template class so that we can show our ability to use templates in the program. Because I created the ball classes, the inventory class, and the obstacle class, I was also in charge of designing the modes and features of the game and making sure each mode is implemented correctly. I was also able to use knowledge gained from my Physics class to make the ball move smoothly and properly by implementing motion equations into the balls' motion. Also since I designed the features that determined the winner of the game, I also broke down the rules and how to implement each ball in the game so that each mode is unique. I implemented a timer class so that the game takes track of time, records and prints the time out, and ends the game after a certain amount of seconds. Furthermore, I helped my partner fix some bugs in the main.cpp file such as inputting/outputting the highscores into a text file and test ran the project multiple times and exposed as many little problems as I possibly could within the game (although I got to admit neither of us were happy when I critiqued some silly bugs and we had to fix them) Alongside the programming aspect of the project, I compiled and broke down the list of topics to cover for the project, designed two of the ball graphics for the game, and helped write a short and simple gameplay page to help explain the rules/features of our Ping Pong game better. I also developed the idea of allocating each topic we covered in class to a bracket reference so we can easily paste the bracket reference onto our code as comments and Quick Find them on Visual Studio 2010 for easy reference.

## Major Bugs Learned Along the Way

A lot of the major bugs involved searching through the entire program just to find a few lines of code that created the bug. Personally, I encountered most (if not all) of the common memory errors created by the Inventory class because it dynamically allocated space for each ball through the STL list template. In the beginning, I did not initialize certain variables correctly such as the iterators so that it would always properly point from the beginning to the end of the list. I received a lifetime error just because I accidentally returned a & or a * and returned a value within the function. There was a lot of object slicing because the Inventory class contained a list of pointers of Balls that pointed towards its derived class. In order for me to fix that I had to properly test its typeid and dynamically casted the function when using the copy constructor and assignment operator. There were memory leaks in the Ball objects, although it was not as bad as another memory leak that happened during review and debugging. My partner and I spent a while finding out why the program would randomly crash, and found out through using Task Manager that the program continuously takes up Memory and CPU usage until it hits the max that the computer can handle, thereby crashing the program. Through discovering that we had to destroy the BITMAP* to save memory, we solved the problem but after a long time of searching through code. I also found out the importance of virtual destructors in the base class (even though it may do nothing) after spending 2 hours and to find out that my program would not properly delete ball pointers that pointed to its derived class (because the inventory's list is a pointer of the abstract, base class, the program crashes by trying to run the destructor on the base class while knowing that the default base class destructor doesn't delete the additional features provided by the derived class.

**What I learned**

Personally, I learned that the biggest and most annoying errors can be caused by the smallest mistakes. I also learned that using a software framework such as Allegro takes time to learn and some bugs can be made just because we didn't know certain aspects of the framework, such as the destroy_bitmap function to save memory space. I also learned that cloud storage such as Dropbox comes in handy to help synchronize files between people and keep the project updated without meeting up in person (up to a point, when the program got too complicated we had problems merging our updates even with Dropbox).